Teaching robots perspective taking

An exploration into Relational Frame Theory and Robotics

By

JOHN BIRKETT

Department of Engineering Mathematics
UNIVERSITY OF BRISTOL
AND
Department of Engineering
UNIVERSITY OF THE WEST OF ENGLAND BRISTOL

A dissertation submitted to the University of Bristol and University of the West of England in accordance with the requirements of the degree of MASTER OF SCIENCE in the subject ROBOTICS within the Faculty of Engineering Mathematics.

SEPTEMBER 2018

Word count: 14650

Abstract

Relational Frame Theory (RFT) is a behaviourist theory which accounts for language acquisition. RFT has considerable implications for cognitive robotics and AI, if correct. One of the core principles of RFT, perspective taking, is explored and modelled in this research using convolutional neural networks (CNN). Data was collected by the author which represented conditions of ownership between a robot and its user to train two types of model. OWNET and OWNET-Object; the former of which classifies just the concept of ownership, the latter attempts to classify ownership at the same time as classifying objects within the image, which CNNs show remarkable capabilities with. The results suggest that ownership alone can be classified with OWNET, but OWNET-Object was unsuccessful in its task. This thesis discusses these results in a wider context, exploring ways this technology could benefit human robot interaction (HRI) in the future. The author believes strongly that the concept of RFT, could have strong benefits within HRI. Further research is needed to explore other aspects of this theory, and build on the work completed in this thesis.

Dedication and acknowledgements

Firstly I would like to thank Dr Manuel Giuliani for the continued support and advice given during this dissertation project. Secondly, I wish to thank Dr Freddy Jackson and Dr Nic Hooper for providing the materials and support needed to conduct this research project. I would like to thank all of them for taking the time out of their schedules in order to attend meetings and answer queries that were had.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: JOHN BIRKETT DATE: 13/09/2018

Table of Contents

List of Tables

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

Our abilities to create robots that can engage on an intuitive level with humans, through social norms, have improved greatly in the past decade. There has been a concerted effort on the part of designers to become more interdisciplinary and connect psychological theory of human interaction when designing autonomous robotic systems. For example, the humanoid robot, Pepper, can keep gaze with a user by tracking their face location and rotating its neck (Soft Bank Robotics, 2017). According to Sibert and Jacob (2000), keeping eye gaze is likely to improve user interaction, based on findings from human-factor experiments. Another example of psychological theory being applied was completed by Mirnig et al. (2017), who found that by using the 'pratfall effect', designers could improve the likability of robots; this involved installing deliberate technical errors in social interactions to make the robot seem imperfect. The current research proposes to draw on a theory from linguistic behavioural psychology, Relational Frame Theory (RFT), and use its principles to develop a robotic agent that understand perspective taking and 'I'/ 'You' relations, according to the deictic frame component of RFT.

1.1 Motivation

According to RFT, perspective taking skills are the corner stone of our ability to communicate with others Jackson and Hooper (2017). It is strongly linked to the ability to empathise and being able to see things through another's eyes, it also accounts for how children learn language. Currently, robots are very limited in their intelligence, whilst computationally quicker in calculation than humans they still are unable to relate past a rudimentary level of strict rules. By modeling this ability into a machine, it is highly likely to prove valuable in allowing for more fluid and adaptive interactions with humans. Ethically this is a difficult subject for psychologists to research practically due to the restrictions on psychological experiments with children. It is also

difficult from a validity point of view. Due to our limited holistic understanding of brain processes, most psychological theories can be challenged by internal validity; is the research testing what it proposes to be testing? Therefore, modeling this behaviour in a robot also in turn would help the psychology community in evidence building for this theory, as it controls for extraneous variables.

If modelled successfully this would be the first time principles of RFT have been recreated in a robotic system this thesis will therefore serve as an exploratory piece into this theme.

1.2 Aims and Objectives

1.2.1 Aim:

The initial aim of this project is to teach a robot to take another's perspective, by being able to take 'I' and 'You' pronouns and reverse them. This project will involve modeling and training a classifier to infer this relationship from a given interaction with a user. If successful it will be able to classify the situation, in a test scenario from an image, and then return the correct prediction of what will come to be defined as ownership, a precursor to perspective taking.

1.2.2 Objectives:

- Research and choose appropriate deep learning model
- Set up experiment: Gain ethical approval, create participant informed consent, write script for interaction with the robot, create efficient way of collecting data
- Undergo data collection process
- Create a model architecture and then train, evaluate and test this model
- Research and discuss possible applications for HRI, including ethical considerations for using such a technology

1.3 Planning

1.3.1 Risk Management

As many of the risks with problems affecting the completion of this project involve booking equipment and ensuring correct environment for software the risk is relatively low. However, a risk management table was drawn up to ensure appropriate measures could be taken to mitigate against risk (See Apendix C).

There was one unforeseen circumstance which could have been controlled for and included in future in a risk register; the time taken for training models. During the training initially a CPU

was used, which took a number of days given the dataset size. When attempting to set up GPU training it was found that the GPU available was not compatible with the software packages being used. Due to this an appropriate GPU needed to be ordered which hindered the timeline of the project.

1.3.2 Timeline

A timeline was created at the start of the project during planning to form a guide for the workflow in the project. This was updated to reflect the true workflow at the end of the project, these can be seen in Appendix D.

On reflection, the original timeline did not take into account the time needed for the retaking of an exam; this wasn't foreseeable as the results came in June for the exams. This did put added strain on the delivery of the dissertation but was managed successfully in reflection. There was also additional time needed at the start for programming the robot and writing the scripts necessary to begin data collection. This was due to a number of unforeseen absences, with key people involved with using that robot being on holiday, so it took longer than anticipated to research operation of the ABB-Yumi robot.

CHAPTER 2

LITERATURE REVIEW

This chapter will cover relational frame theory in more depth by providing the background in the psychological literature, its relevance, and a wider review of the literature from this theory (2.1). Finally the last section (2.2) will cover similar techniques used in the context of robotics, this is the first time RFT has been exclusively looked at in the context of practical robotic applications; so 2.2 will review the work which has similar relevance to the current thesis.

2.1 Relational Frame Theory

This section will contain a background (2.1.1) to the theory of relational frames and the core concept behind the theory (2.1.2). It will also review the relevant literature in the psychological literature. It will conclude with a recommendation of a narrow area of the theory to focus on as the theme of this thesis.

2.1.1 Background

A fierce debate within psychology and linguistics has been ongoing for decades over a simple question; is language learned or innate? With behaviourists arguing that it is completely learned and others arguing it is completely innate. The main criticism being that the young children often apply their words and create new phrases which have not been learned. Relational frame theory (RFT) attempts to account for this by suggesting that there are some simple rules we teach to children in the first few years of development; such as the relationships between two objects in their size or distance between one another (Jackson and Hooper, 2017).

The original theorisation of behavioural and analytical acquisition of language was proposed by Skinner (1957). Skinner used the idea of the operant [1], which at the time had been well researched in animal studies, as the underpinnings for this behaviourist understanding of language acquisition; the theory suggested that language was completely learned, through social interaction, and reinforced both positive and negative social interaction. Huge criticisms were made against this theory due to its lack of explanation for complex and novel human behaviour, and in this case novel sentence construction in children (Chomsky, 2002). The problem in the early work done by Skinner, was a large assumption that behaviour acquisition in mammals would work across different species, Skinner applied this to all behaviour and assumed that language was included in this. Successive experiments, for example Galizio (1979), found difference in behaviour acquisition between human and rats; with humans being able to perform better under rule-based behaviour tasks. Behaviours are not all equal and some are not rule based, despite seemingly like they are. For example, Estes and Skinner (1941) tried to get rats to learn fear, but this early work did not take into account is that the rats were already fearful in the settings he put them in, according to Chomsky (2002). So the rats were not necessarily learning a rule that a stimulus is equal to fear and to produce a response rather than the rat is already hard wired to be fearful.

Another issue in language, is it is generally a bi-directional behaviour undertaken between humans, rather than simply by humans individually, usually in the form of spoken word (Houwer, 2013). There was fair criticism of the behavioural analytic theory up until this point due to these short comings. Sidman (1971) found evidence of a phenomena which he coined 'Stimulus Equivalence', in which words that were printed, spoken, and within images were able to be mutually substituted. For example, 3 equal variables exist, X means Y and Y means Z, the participant (in this case a developmentally disabled child) was able to use relationships to infer that X also means Z. Sidman suggested that the current behavioural theory could not account for this, concluding that stimulus equivalence consisted of three properties: 'indentity matching', picking X whilst X was shown; 'mirroring', picking X when Y was shown, after learning that X means Y; and 'transitivity', picking X when Z is shown after inferring the relationship from the two rules.

This generalisability, in the rule-based phenomenon of stimulus equivalence, was able to provide the missing parts to traditional behaviourist theory that was needed to learn language. An account of the link between equivalence and language was provided by Hayes (1991) which led on to the grounding of RFT as a general theory. Hayes argued that the relational frames should not be confused as the products of logic, but what gives rise to logic; this Hayes explained was due to the fact that arbitrary things and events, are given relationships by engaging with a language community, logic occurs as a product of the relational frames, and is a social consequent. From a social constructionist perspective this is true, but it comes down to philosophical interpretation.

[1] The Operant is a term which refers to a class of different events which trigger a response, these responses are seen as members of the class and can be modified through consequence; the idea of positive and negative reinforcers

2.1.2 Theory

In 2.1.1 the background to RFT was discussed. This subsection will now cover the core concepts in RFT before concentrating on one aspect in particular for the rest of the thesis.

RFT, as proposed by Hayes, focused on what was coined "Arbitrarily Applicable Relational Responding" (AARR). This encompassed the findings of Sidman, the three elements defining stimulus equivalence referred to earlier in the earlier section of this review. AARR essentially focused on the context surrounding the relations between objects/events, this is distinct from physical attributes such as size. For instance, cricket balls could be roughly the same size as a grapefruit, but a contextual property is one is heavier than the other, once we learn this we can infer other relations from that frame. This is where the theory attempts to explain higher human cognition and language, by way of objects and events possessing multi-dimensional frames which can be arbitrary or non-arbitrary.

A good example of why this is a generalisable account for language was given by Houwer (2013). A similar example to what they suggested could be; presenting a child with a contextual word 'taller', if told of three stimuli person X is taller than person Y and person Y is taller than person Z. They can derive, from this contextual word, that person X is taller than Z without physically seeing the stimuli. This works the same if the stimuli, instead of a person, becomes buildings, the stimuli are irrelevant to the contextual cue relating the stimuli. Why is this important? Studies have found that this is a uniquely human ability, so far (Hayes, 1989). Associative learning, for example operant conditioning, has been found to be capable in non-human subjects, this means that they can learn behavioural responses localised to the stimuli they have been trained on (Dube et al., 1993). Generalising a learned response to new stimuli is difficult, even our closest living ancestor, the chimpanzee, is unable to produce the correct response and generalise when new stimuli are presented according to Yamamoto and Asano (1995). RFT is arguing against this associative type of learning, which is also the traditional Skinnerian view of behaviour acquisition.

The implications of this theory being accurate are important, as it accounts for several distinct variances that separate humans from other animals in terms of logic and reasoning. Over the past couple of decades researchers have examined the inferences of RFT including within domains such as: mental pathologies such as OCD (Nicholson and Barnes-Holmes, 2012); understanding idioms and metaphors (Stewart and Barnes-Holmes, 2001); assessing implicit beliefs of sexual offenders (Dawson et al., 2009); and offering explanations of emergent frames in early child development (McHugh et al., 2004). The evidence base is still growing largely in support of the theory, with no empirical studies, at present, contradicting other empirical RFT findings. A citation analysis conducted in 2017 found that most studies conducted are published in reputable journals (O'Connor et al., 2017). O'Connor and colleagues found 521 papers were cited between 2009 and 2016, with 55.3% being empirical and 44.7% being non-empirical. Non-empirical studies included reviews and conceptual papers, the conclusion being that there was evidence of growth

in RFT empirical research. This conclusion was drawn from a previous analysis conducted in 2010 showing similar conclusions between the period of 1991 and 2008 (Dymond et al., 2010).

There are nine different dimensions of frames, that relate between objects and events, that are central to RFT according to Houwer (2013):

1. 'Co-ordination' – (e.g. human is also person)
2. 'Distinction' – (e.g. man is not the same as woman)
3. 'Opposition' – (e.g. man versus animal)
4. 'Comparison' – (e.g. that human is smaller than this human)
5. 'Spatial' – (e.g. that human is on the right and that human is on the left)
6. 'Deictic' – (e.g. from the perspective of the individual assessing, I am in front of person A but behind person C, which is different from purely spatial)
7. 'Temporal' – (e.g. I ate breakfast after I ate yesterday's dinner)
8. 'Hierarchical' – (e.g. a human is a type of mammal, which is a type of Mammalia)
9. 'Causal' – (if I eat too much, I will put on weight)

This thesis will focus on the sixth item, deictic frames, in particular it will look to model this frame in order to improve human robotic interaction (HRI). One element of this particular frame is that there is an 'I' and a 'You', and that these are interchangeable depending on the perspective being taken. Humans learn that I isn't tied to a particular event or stimuli, they learn that it is the personal perspective of the person using it. Likewise, when 'You' is used, humans learn that it refers to another person/agent from the perspective of an 'I'. There are two elements to this the author feels is important in being able to use this perspective framing effectively; there is the recognition of a stimulus or event, and also its relationship to the individual, or ownership of that event or stimulus. In an autonomous agent this would equate to object-recognition and ownership classification. This thesis will attempt to build a model which can classify ownership.

2.2 Related Work within Robotics

The frames discussed above, in some ways, have long formed foundations of computational logic and practiced in autonomous systems, albeit without the conceptual theory of RFT behind it. For example, in the past twenties years machine learning has enabled the recognition and classification of objects within images (Lowe, 1999; Girshick et al., 2014), this in turn allows for some relational co-ordination, distinction, opposition and comparison which can be made from preprogrammed responses to detections.

Sensors were using primitive versions of the principle in 1997 for obstacle avoidance in wheelchairs (Katevas et al., 1997). Systems involving neural networks typically rely on a lot of data to be fed into them to be able to classify objects, which is not how human infants learn how to classify things. Some, such as Greenway et al. (2010), have argued that RFT could be the answer to the general theory of consciousness, although this is doubtful, it still will be worthwhile attempting to model aspects in a machine. Especially since we have rudimentary versions of other frames already, such as 'Causal', within most programming languages. This includes all programmatic responses, which hard coded through internal logic statements such as "if this then that".

One attempt at replicating the general 'stimulus equivalence' phenomena, discussed in 2.1.1, have been made in neural networks by Okada et al. (2005). Okada et al. provided a model for the symmetry problem, demonstrating the co-ordination frame, and successfully tested its validity. The important distinction was that they were able to use an inverse model on a trained neural network. The input to the network was pictures of 'Bananas', which was trained to be the labelled to the word 'Banana'. This relationship was able to be reversed using the 'Iterative Inversion Method'. This was different to the current goals of this thesis, whilst RFT is based on early stimulus equivalence research, the co-ordination frame is useful to have modelled. Okada et al.'s research demonstrated the ability of neural networks to be inversed, and use their trained output label as an input to recreate the pixels in an image of a Banana; this is showing that A=B and B=A essentially. There is no other pieces of research following on from this work, it would be useful to explore this further in an updated piece on the subject as this was originally completed in 2005 and machine learning has advanced in the past thirteen years.

A paper from Google's Deepmind discusses the closest likeness to relational frame theory in the context of machine learning. Santoro et al. (2017) discusses a module they have created which does relational parsing between objects within images. In order to answer relational questions, such as "What is the color of the ball next to the brown cylinder?", which the network handles with ease according to their results. This is different from the current research in that it is not concerned with perspective taking between agents, just relating between objects.

Conclusion

This chapter has reviewed the background and current literature into relational frame theory. It has also discussed this in relation to robotics. The relevant machine learning and robotic literature has been reviewed, with only a small number of similar pieces of research. There are currently as of writing this thesis no other works examining the role of relational frame theory in robotics or machine learning. The author feels that this is an important area to explore. Therefore the direction of the thesis will be that of creating a model which can classify ownership as a step in inferring the perspective of an agent in relation to an event or stimuli. The next chapter will explore possible methods of completing this task.

CHAPTER 3

METHODS

Following the last chapter, the problem which will be specifically addressed throughout the rest of this thesis is that of a precursor to perspective taking, classification of ownership. The first section of this chapter will focus on deciding the appropriate machine learning method for this problem. The second section describes the data collection process to present labeled data to the machine learning algorithm for it to learn from. The final section describes the architecture and training algorithm for the final models presented by the thesis.

The machine used during the training phase was run on a Windows 10 operating system with 8 GB of DDR3 RAM and 240 GB SSD storage; the processor was an AMD FX-8350 Eight-Core Processor (4.00 GHz) and the graphics card was an 8GB GDDR5 NVIDIA GeForce GTX 1070 Ti which has 2432 Cuda cores [1].

All of the programming completed during the methods section was completed using 'Python 3.6', copies of the code used in this section can be seen in Appendix's F, G and E. The machine learning packages and modules used with python during this thesis were: Tensorflow (Abadi et al., 2016) and Keras (Chollet, 2016). Whilst other python packages were used for image and data processing/visualisation, these included Numpy, Pillow, OpenCV, Matplotlib. The descriptions of their purposes and a full list of functions used can be seen in table 3 below.

3.1 Machine Learning Methods

There are typically three learning methods in machine learning; supervised learning, unsupervised learning and reinforcement learning. Supervised learning is a process of mapping input space to a given output space using labeled data (Floreano and Mattiussi, 2008). Unsupervised

[1] Cuda Cores are a technology NVIDIA provide which allows programming packages such as 'Tensorflow' to directly access the parallel processing power of a graphics card

Package Name	Purpose	Description
Tensorflow	Machine Learning	This package is an opensource machine learning library which allows for high performance computation using GPU's. It provides a stable platform for building machine learning algorithms in python.
Keras	High-level API for Tensorflow	This is another opensource library written in python, it runs on top of Tensorflow, and others, and makes building an training neural networks much simpler.
OpenCV	Image Processing	An open source library with classes and functions which highly aid computer vision. The library provides support for Tensorflow.
Pillow	Image Processing	PIL is a imaging library which adds support for opening files of different formats.
Matplotlib	Plotting	A plotting library which works with Numpy
Numpy	Scientific Computation	Numpy is a library which adds support for processing multi-dimensional arrays and other complicated mathematical functions on them to speed up processing speeds.

Table 3.1: Programming libraries used during this thesis

learning is essentially a method of clustering unlabeled data. Finally, reinforcement learning is a way of having agents make decisions based on past behavior through reward maximization, essentially the algorithm attempts to maximize some reward metric (Floreano and Mattiussi, 2008).

Both supervised and reinforcement learning are the most similar to human learning. Category learning (Bruner et al., 1956), a similar concept to supervised learning, is a theory of how humans are able to sort attributes into sets, a form of classification; this would make supervised learning an appropriate method to use in this thesis's goal. Reinforcement learning is similar to operant conditioning (Skinner, 1957), referenced earlier in Chapter 2, which uses positive reinforcers' (a reward) to alter behaviour. At this stage, reinforcement learning may not be an appropriate method to use as it focuses more on behaviour of an agent that can already understand its environment.

As a problem, creating a system that can solve the question of ownership of an object, where there are discrete categories, is that of an input-output classification one; in this case there are two categories, robot or user. Therefore supervised learning is the appropriate method to use for the problem of ownership classification.

In statistical classification problems, the task is deciding, given a current observation/ example, what sub-category that example belongs to, it can also be thought of as pattern recognition (Floreano and Mattiussi, 2008).

It is useful to know what data is going to act as input to a classifier system. For perspective taking, it would be useful to get as much information to make decisions as possible, in a multi

modal way; for example from image, depth and speech. For this thesis, it is hypothesised that for classifying ownership in an observation it will be enough to use image data alone.

3.1.1 Brief background of Artificial Neural Networks

In 3.1, the various methods of machine learning where considered, it was decided that this thesis should use the supervised learning method. For this artificial neural networks will be explored for a solution to this thesis's problem.

Artificial Neural Networks (ANN) are computational models that implement the behaviours of biological neuronal activity to produce efficient processing. They begin at their primary building block, the 'Perceptron', which has an arbitrary number of inputs, a threshold, an activation function and some singular output value (Rosenblatt, 1958). An ANN is typically characterised by an input layer of data points followed by a hidden layer/s with some number of nodes and finally an output layer of nodes; the number of which depends on the problem, in classification problems this would be the number of labels, in regression problems the number of independent variables (Floreano and Mattiussi, 2008). All of the nodes within ANN's are normally fully connected, however experiments have tested the effects of partially connected networks (Chang, 2012). There are many varieties of architecture but the most common used for linear and non-linear problems is the Multi Layered Perceptron (MLP) (Figure 3.1).

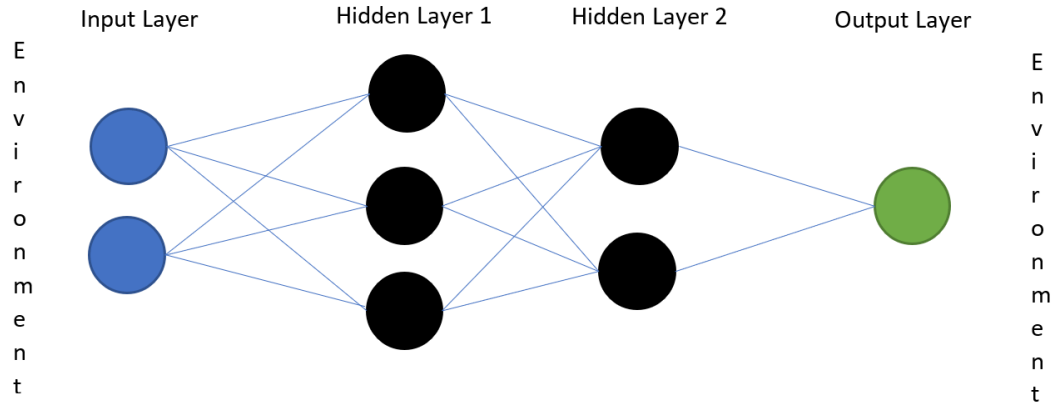


Figure 3.1. Diagram of a generic Multi Layered Perceptron Artificial Neural Network

An individual neuron (y_i) can be described mathematically as the net sum of its inputs (X_j) after calculating the dot product of its inputs and the respective connecting weights (W_{ij}), followed by subtracting a threshold (ϑ). The threshold mimics the biological mechanism, an 'action potential', which requires a certain amount of energy for a neuron to fire. Finally, it is then multiplied by an activation function (ϕ), the equation can be seen below in 3.1.

$$\boldsymbol{y_i} = \phi\left(\sum_{J=1}^{N} W_{ij}x_j - \vartheta_i\right) \tag{3.1}$$

'Learning' in ANNs is completed by updating connection weights between layers. This is typically achieved through the back-propagation algorithm (Floreano and Mattiussi, 2008). The purpose of back-propagation is to calculate the partial derivatives of an error function with respect to each weight in the network. This when applied throughout the network aims to reduce the lowest possible error function through the concept of gradient descent [2]. The amount of epochs [3] is usually determined by some early-stopping criteria, such as when the the validation dataset error increases or some maximum number of iterations is reached.

MLP's have been described as, and proven to be, universal function approximators according to Hornik (1991), a quality due to the multiple layers of connected nodes rather than the activation functions used. This means that for any given function there exists a neural network with the appropriate architecture that can approximate from any possible input, its given output. This property has made them adept at solving a variety of problems; object recognition (Mercimek et al., 2005), predicting cardiovascular disease risk (Weng et al., 2017), inverse kinematics (Köker et al., 2004), and even in predicting price changes in bitcoin (Jang and Lee, 2018).

Whilst MLP architectures have been successfully applied to image classification problems, they can experience the curse of dimensionality due to their fully interconnected nature (Krizhevsky et al., 2012). For example, a 20*20*3 (20 pixels in height, width and 3 color channels) in just the first hidden layer would require 1200 weights; A 150*150*3 would require 67,500. Also these network designs do not account for spatial nature of image data, they treat pixels in close proximity to one another the same as those much further away (Siciliano and Khatib, 2016).

3.1.2 Convolutional Neural Networks

In section 3.1.1, we discussed a brief background and common model used in ANNs, it was also discussed why these were not completely suited to image classification. 3.1.2 will discuss convolutional neural networks which have been successful in image classification techniques, their architecture and foundations in biology will be discussed too.

Convolutional Neural Networks (CNN) are a type of deep feed-forward ANN, which are very capable at spatially invariant problems as they hierarchically abstract feature representations through local operations. CNNs are a variant of the MLP architecture and include in their final layer an MLP to calculate the final output. They have proven particularly capable in various image recognition tasks (Skin Cancer prediction: Esteva et al., 2017; Facial expression and

[2] "This draws interesting parallels to how children learn concepts such as ownership. For example, a parent giving a child an Apple whilst telling them "you have an apple", the child replies "you have an apple", then the parent corrects and explains "no, I, I have an apple".

[3] Each cycle of training, where all examples have been ran through the network and back-propagation has occurred.

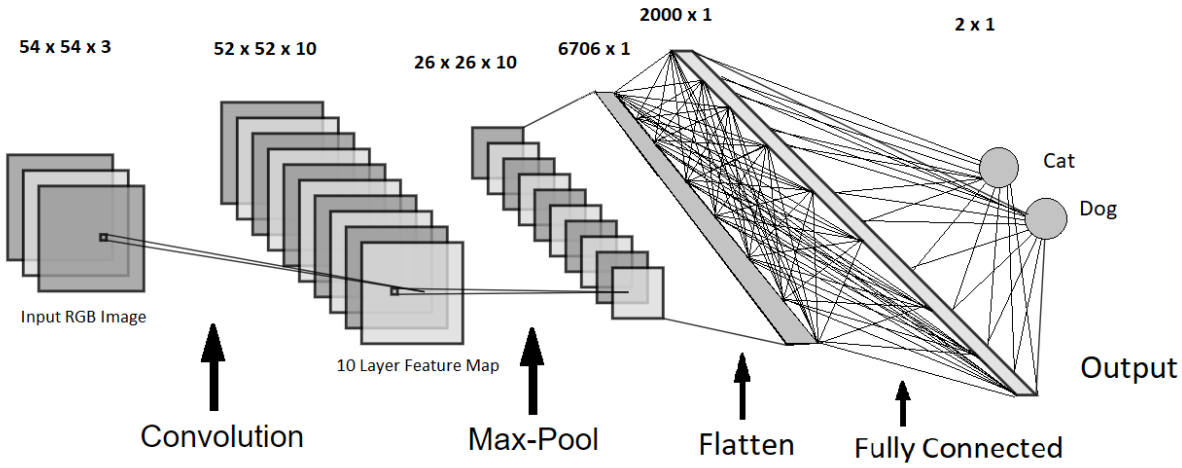


Figure 3.2. Diagram of general Convolutional Neural Network Architecture

emotion classification: Jorda et al., 2017), but have also been useful in areas such as Natural Language Processing (Sentence parsing: Kim, 2014; Audio classification: Hershey et al., 2017).

The architecture (see Figure 3.2) of a simple CNN for image classification consists of a 3-dimensional input (height, width, color channels) followed by a convolutional layer, pooling layer, ReLU activation layer, fully connected layer, and and an output layer. The next set of subsections (3.1.2.1 - 3.1.2.6) will cover these layers in more depth, and why they relate so closely to human biology.

3.1.2.1 Convolutional layer

The convolutional layer involves taking a small kernel filter (for example 3*3*3) and moving it along the entire image, at every stride (movement) performing a dot product between the kernel and sections of the image's pixel values. These kernel filters are equivalent to the trainable weights in regular ANN's, a 3x3x3 kernel each have 27 trainable weights. This computation produces a single scalar value in the next layer which forms a feature map, the size of which is the sum of all the possible unique positions the filter can be put on the image (Siciliano and Khatib, 2016). The entire layer is in fact compromised of a set of independent convolved feature maps, the number of which is arbitrary, and often involves trial and error process to find the right number. All of the kernel filters used for each feature map are initialised randomly (Buduma and Locascio, 2017). Each node/neuron in the feature map is connected to a small section of the input, this enforces local connectivity and allows CNNs to exploit local spatial correlations, which is different from fully connected networks like MLP's.

Local connectivity between neurons and their input allows for parameter reduction, reducing

the curse of dimensionality, optimisation problem, when working with high dimensional spaces, such as images, according to Buduma and Locascio (2017). This size of the local connectivity is a hyper-parameter for the model, known as the receptive field of the neuron, equal to the size of the kernel filter.

Three further hyper-parameters control the shape of the convolutional layer: zero-padding, stride and the number of filters. The number of filters or feature maps is equal to the depth of the layer, each layers neurons 'learn' to activate for different features in the input to the layer such as edges or color. The stride controls the width and height of each feature map, when the stride is 1 the filter moves 1 pixel at a time, when it is 2 it moves 2 pixels. This is normally no more than 2 as it can cause less overlap between the receptive fields. The zero-padding parameter is where zero's are added to the edges so that the spatial dimension of the input is preserved (Buduma and Locascio, 2017).

An assumption that underlies this layer is that the kernel filter weights, if good enough for one section of the image, are useful on the whole image. This means that the amount of free trainable parameters for each neuron per feature map is the the size of the kernel (e.g 3*3*3 kernel has 27 trainable weights per feature map). This is in contrast to a fully connected MLP which would have trainable weights, for each neuron, a number which is equal to the size of the input space (e.g 100*100*3 image would have 30,000 trainable weights for every neuron in the first hidden layer). This increases training speed by reducing the free parameters and also resolves the vanishing gradient problem according to (Siciliano and Khatib, 2016).

3.1.2.2 Pooling layer

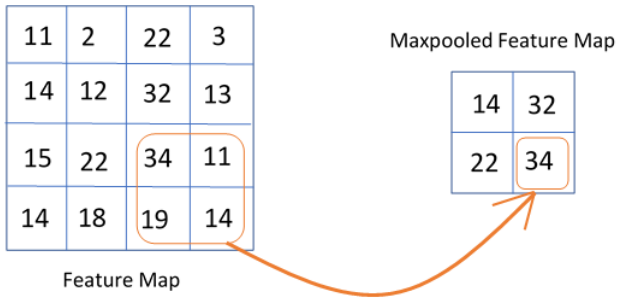


Figure 3.3. Diagram of Maxpool operation, the 2x2 kernel is passed over the feature map in strides of 2, selecting the largest pixel value to copy to the pooling layer

After a convolutional layer, a pooling layer is added, its function is to reduce the size of the network by lowering the resolution; this helps to restrict the amount of parameters and introduces further non-linearity. The approach which is generally used, and will be used later in this thesis is 'Maxpool'. This method uses the maximum value from a cluster of neurons in the

previous layer (the cluster usually being 2x2) and map this value to a neuron in the pooling layer. This can be thought of as partitioning of the previous layer into sections, see Figure 3.3. This occurs independently on all of the feature maps, so the number of the depth of the previous layer remains the same.

3.1.2.3 Rectifier Linear Unit layer

As the convolutional layers are not fully connected, activation functions are used after each layer to increase non-linearity without affecting the receptive fields of the neurons.

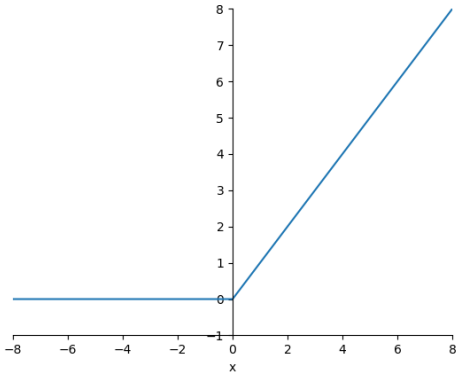


Figure 3.4. Diagram of a ReLu function.

$$f(x) = Max(0, x) \tag{3.2}$$

Xu et al. (2015) found that Rectified Linear activation functions (ReLu) were far more efficient for forward computation through CNN networks as opposed to more traditional activation functions such as a Sigmoid. Mathematically, a ReLu function can be described as seen in Equation 3.2, a visualisation of this function can be seen in Figure 3.4.

3.1.2.4 Fully connected layer

In principle the second to last layer acts like a hidden layer from an MLP (Buduma and Locascio, 2017). This layer is fully connected (FC) with weights to the previous neurons (which are also learned) from the last pooling layer and provides a flow of the whole image to the single output neurons; the mixing of signals provides any classification node with information from every pixel in the original image.

To add this layer the network at the final pooling layer, all of the neurons, within all feature maps, must be flattened into a 1D Array. After this, the features in this array have a weight kernel, bias and activation function passed over it to form scalar values for the FC layer.

3.1.2.5 Output layer

Finally the output layer is where the classification occurs, much like in MLPs, and is therefore equal to the number of independent classes. The output layer is also known as the loss layer, and is used for calculating the loss (error) function of the network. The loss function is how the networks training parameters know to correct for deviations from the predicted labels and true labels values. Finally weights, biases and an activation function fully connect all of the previous neurons in the FC layer to N output neurons. The activation function for this final layer is based upon the type of problem and classes being trained for. The two most common are 'Softmax' and 'Sigmoid'. The former is used where classes are mutually exclusive and outputs probabilities of the class being present in the output neurons, all of the probabilities add to 1 (e.g [dog = 0.2, cat = 0.8]); the latter is used as binary indicator of whether a class is predicted or not and does not add to one, therefore is used when classes are independent from one another.

During each epoch, a loss or error function is calculated on the sum of the outputs for that epoch. This loss function provides a single scalar for the entire epoch in order to control the size of weight changes in each layer during the back-propagation. Classification tasks usually utilise cross-entropy, or categorical cross-entropy loss; the latter of which is used in this thesis's experiments.

3.1.2.6 Biological Significance

CNNs were built with biological inspiration, possibly more so than traditional neural networks, as they are based on theories of the way animals process incoming photon information in the visual cortex (Rolls and Deco, 2002). Inside the visual cortex (VC) exist cortical neurons which selectively respond to stimuli/ input from specific regions of the visual field (VF), these are referred to as 'receptive fields'. This is where the architecture of CNNs as described in the above section begins to fit with biology.

Hubel and Wiesel (1962) showed this within cats VC, where individual neurons directly responded to small areas of their VF. Furthermore, the authors showed two types of VC cell; simple cells which were stimulated increasingly by straight edges with certain orientations in their local receptive field; complex cells that were larger in their receptive fields and were not sensitive to edge position.

Directly relating this to CNNs is straight forward; simple cells are emulated by the convolutional layers, performing feature detection, whilst complex cells are emulated by the pooling layers which assist with the invariance to spatial location of features within the image according to Scherer et al. (2010). The Neocognitron, a precursor to the first CNN architecture, emphasised this more in the design of the networks' layers, naming specific layers Sn and Cn in reference to the cells. (Fukushima and Miyake, 1982). In particular, the spatial invariance is an important feature of CNNs, in a traditional MLP you would need training examples of a feature in all

possible positions within an image to generalise, a CNN can abstract this from examples in one position of the image.

3.2 Data Collection

In the previous section, convolutional neural network architectures and theory was explained. In this section the author describes the collection process of data necessary for training the OWNET model, a single ABB-Yumi was recorded, from the robots perspective, interacting with a user. A diagram of the hardware setup is shown in Figure 3.5. Subsection 3.2.1 will discuss the hardware setup used for data collection. 3.2.2 will discuss the ethical approval, informed consent and other details in regards to participants used in the collection process. Finally, 3.2.3 will discuss the procedure of undertaking the data collection.

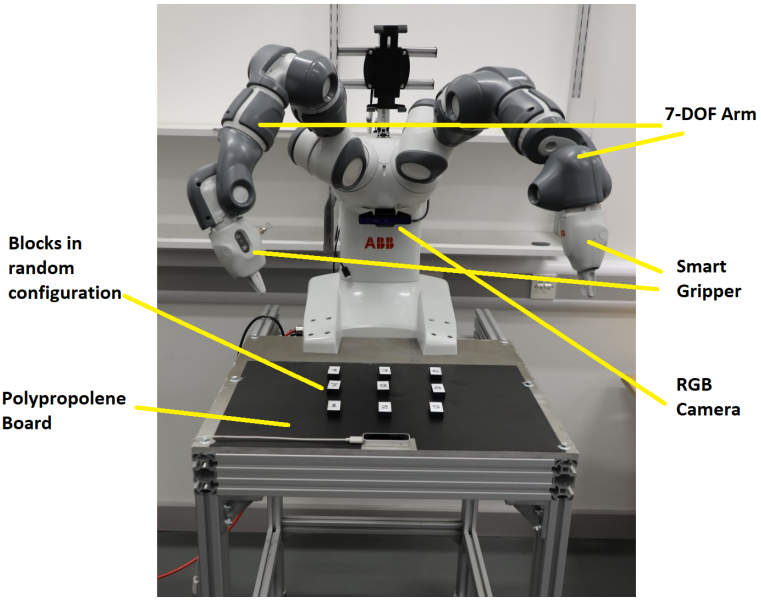


Figure 3.5. Robot and hardware setup

3.2.1 Hardware

The experiment involved the use of a compliant collaborative robot, the ABB-Yumi, which is equipped with two 7 degrees of freedom arms and two IRB 14000 smart-grippers. Attached to the robots center was a BlasterX Senz3d camera which was used to capture 600x400 RGB and depth video at 30 FPS of participants interacting with the robot, although only the RGB data was used for training OWNET.

In order to improve image contrasts and reduce reflection from the steel surface a matte black polypropylene sheet was measured and cut with the laboratory's laser cutter. Nine labels for

the blocks were also engraved with the cutter and used to standardize the training process. The blocks were arranged in a 3x3 grid formation and each numbered block was captured in at least every location once in each of the two sets of ten trials. This was standardised by moving each block to the left, from the human perspective, with the final block at the bottom left being moved to the top right. This was completed nine times with the final time the blocks being placed in random order on the board. This is illustrated below in Figure 3.6.

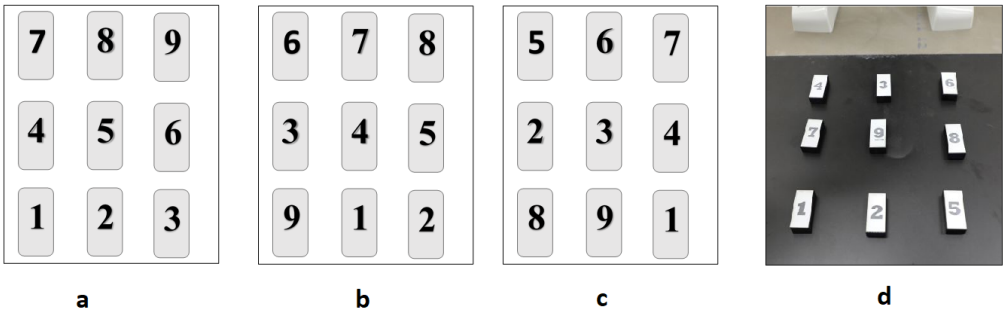


Figure 3.6. Board configurations between each trial: a represents the first trial, after this, in b, each block is moved to the next consecutive position, and then the same again in c. d shows a real image of one of the board configurations

The robot was manually jogged into locations using the systems on-board flex-pendent saving the position of joint angles in 12 sequential arguments. The robotic arm was programmed to move at constant speed of v20. Within each sequence the gripper was programmed to interact in one of two conditions, grasping or touching. Each arm had six sequences of interactions per trial; the arms corresponding edge blocks being visited twice and the middle blocks each being visited once by each arm. The control code can be seen in appendix F for both the left and right arm.

In order to capture the data needed an open source computer vision library was used with python, 'OpenCV2'. The full code for capturing the data can be seen in Appendix G. Board conditions were stored in lists and accessed based on the experiment number (e.g Experiment 1: [1,2,3,4,5,6,7,8,9], Experiment 2: [9,1,2,3,4,5,6,7,8]...). OpenCV was used to capture data for both human and robot interactions for a total of 30 seconds. This equated to 900 frames per interaction (30 x 30), with 12 interactions from the robot and 12 from the user this equated to 21,600 unique examples to use, from each trial, in the training of OWNET [4].

3.2.2 Participants

Ethical Approval was obtained using the universities 'Ethical Review Checklist' and obtained a low risk rating and was approved by the Supervisor of this project, Dr Manuel Giuliani. A copy of this form can be seen in Appendix A.

[4]This was reduced to 2700 unique images per trial instead, this was after the similarities between each frame being to high, after this frames were taken from footage at 8-frame intervals

Twenty participants, each with individual trials, were involved in the data collection process. Normally for HRI experiments it would be sensible to collect data from participants whom have never interacted with a robot. The effect of novelty for this experiment was considered, as the image data only contained lower torso and hand information, any effects were predicted to be low, if at all present. Given this, participants were gathered within the Bristol Robotics Laboratory exclusively over the data collection period.

Participants were all given fully informed consent which can be viewed in Appendix B. The intended use of the data being collected was fully explained before hand along with the wider research aims. It was explained that they had the right to withdraw their data within two weeks of taking part in the experiment and given instructions on how to do this (through use of personal code which was attached to their trial number). It was also made clear that no personally identifiable information was required to be stored for the experiment. Participants were shown examples of image data prior to the task so that they were aware of what was being captured. Finally, each participant was made aware of risks from interacting with a high powered robot and were given explanations as to how these risks had been reduced.

3.2.3 Procedure

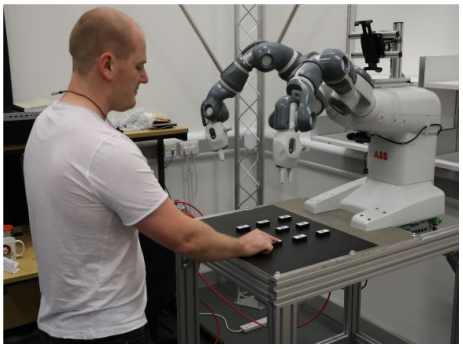


Figure 3.7. Participant interacting with the ABB-Yumi robot during a trial

Participants were given a time to attend the data collection after agreeing to take part. As there was no need in this data collection process to withhold information from participants they each received a full explanation of the research aims. Participants then had the chance to ask any questions they had.

After this, instructions were given explaining how the flow of the data collection would proceed. It was made clear that they would take turns with the robot in interacting with blocks on the platform and that the robot would go first. Participants received instructions of which hand to use before each turn, this also which block to interact with, ensuring complete coverage of the local area of the board. An example of an instruction would be "Right hand, touching block

eight". See Figure 3.7 and 3.8, to see both a participant and the robot interacting with one of the blocks.

Figure 3.8. ABB-Yumi interacting with a block during a trial

At the beginning of every interaction the program was triggered to capture the 30 seconds of data, for both the robot and participant. This would then be labeled for who interacted/owned the block; 'R' indicated the robot and 'U', the user. The label would also include information on which block was being interacted with (e.g 'R-9' or 'U-4'). As the research aims had been fully explained there was no need for a debrief, however, participants were asked if they had further questions regarding their interaction.

After each trial, blocks would be reordered as described in the above section.

3.3 OWNET

The previous section explained the hardware, participants, and procedure used to collect data that can be used to train OWNET. Section 3.3 covers the data preprocessing, architecture and training of OWNET. The preprocessing and organisation of the data will be explained in 3.3.1. The models proposed architecture will be described in 3.3.2, as well as attempts to prevent over-fitting. The training process will be walked through in 3.3.4, with the techniques and optimisation algorithm explained.

3.3.1 Data Preprocessing

Stills from the video files, showing the interactions between the robot and participants, were converted into jpg files with the same dimensions as the video (600x400) using *OpenCV*. They retained the labels captured during the data collection process ('R-1....', 'U-3....'), adding the frame that they came from and the trial number. At this point two of the trials footage was corrupted and discarded. Four whole trials of data were separated at this point to form the validation and

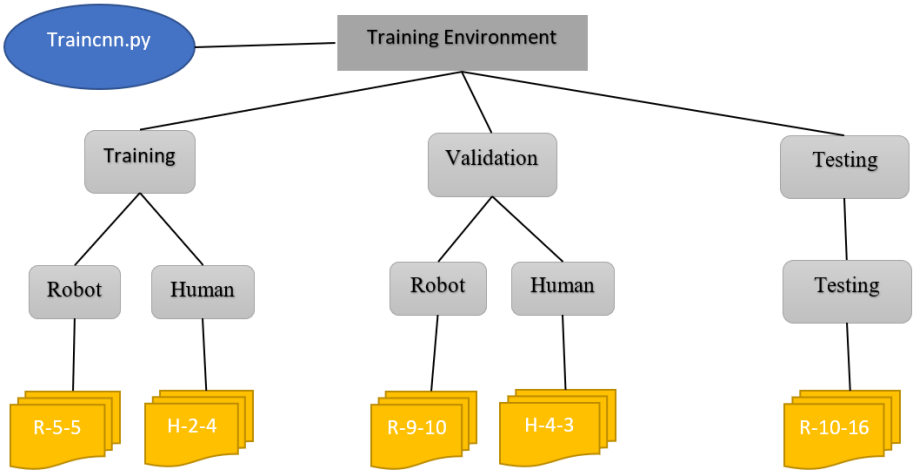


Figure 3.9. Flowchart of folder structure

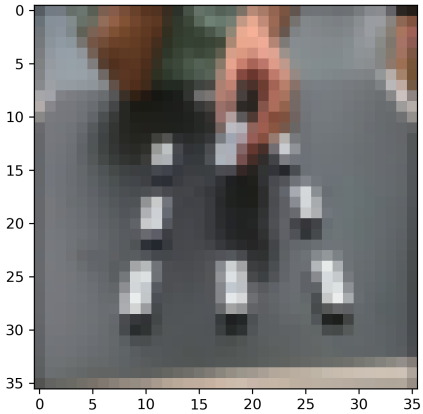
testing set. To ensure the model has images which are completely novel to it. Data is split in this way so that the model can test to see if it is still able to generalise to data it has not seen; so it has some reference of when to stop training early if it is beginning to be trained too well.

These files were then rearranged into a folder structure to enable efficient training, see Figure 3.9 to see this structure. The images are then read in each iteration of the training using *Keras's* "Flow_from_directory" function in its Image_Data_Generator class. At the same time this function can resize images, this was for two reasons: firstly, it reduces the size of the network which improves training speeds, allowing training in batches; secondly, the computational requirements become larger the higher the image size. Resolutions in past research have ranged from 32x32 (Eigen et al., 2013) to 256x256 (Krizhevsky et al., 2012). For this project, two resolutions were chosen for the OWNET network to train and compare, 36x36 and 72x72, see Figure 3.10. In all of the training conditions RGB images were used, this increased the size of the network and the training time; however, as the networks goal is to classify ownership between robot and human, the colour differences between skin and plastic were considered to be the most likely significant features.

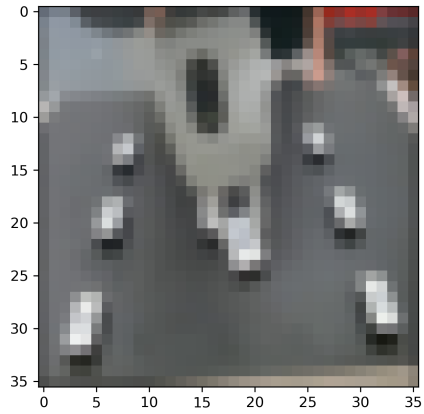
3.3.2 Model Architecture

The architecture used in this thesis was based on the architecture used in the ImageNet paper (Krizhevsky et al., 2012). The architecture was modified to suit the problem in this thesis and the computational abilities of the hardware training the model. The network contains 3 convolutional layers and 1 fully connected layer which is connected to an output layer consisting of a 2-way softmax activation function. The amount of feature maps was manipulated for each layer to increase in multiples from the first layers number. Two initial starting feature map numbers

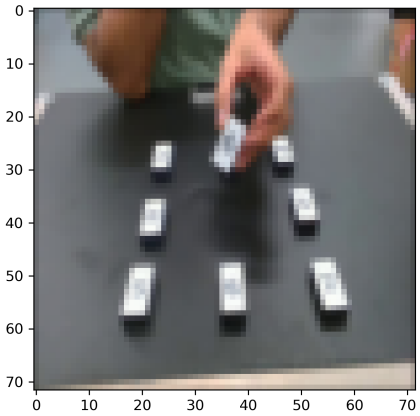
(a) Human owning situation 36x36



(b) Robot owning situation 36x36



(c) Human owning situation 72x72



(d) Robot owning situation 72x72

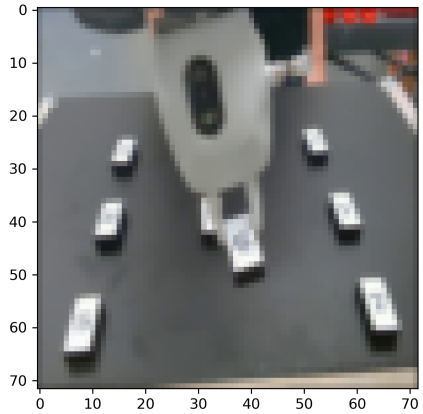


Figure 3.10: Showing examples of different resolutions used during training of OWNET

were used (6,30) for both image resolution conditions.

The first of the convolutional layers receives the input image and filters the image with 6 or 30 kernels of size 3x3x3 with a stride of 1 (providing maximum overlap in receptive fields within feature maps). The second layer takes as input the result of the previous and filters it with 12 or 60 kernels of sizes 3x3x6 and 3x3x30 respectively. The final convolutional layer passes 18 or 90 kernels of sizes 3x3x12 and 3x3x60 respectively. Each of the layers contains ReLu functions followed by a Maxpooling operation which has no overlap. Figure 3.11 shows an annotated diagram of the network, with input size 36x36x3 version and initial feature map size of 6.

In all of the training conditions, after the final Maxpooling operation, the network is flattened and connected to 650 fully connected neurons. A full list of the model architectures that were tested and results reported on can be seen in Table 3.2.

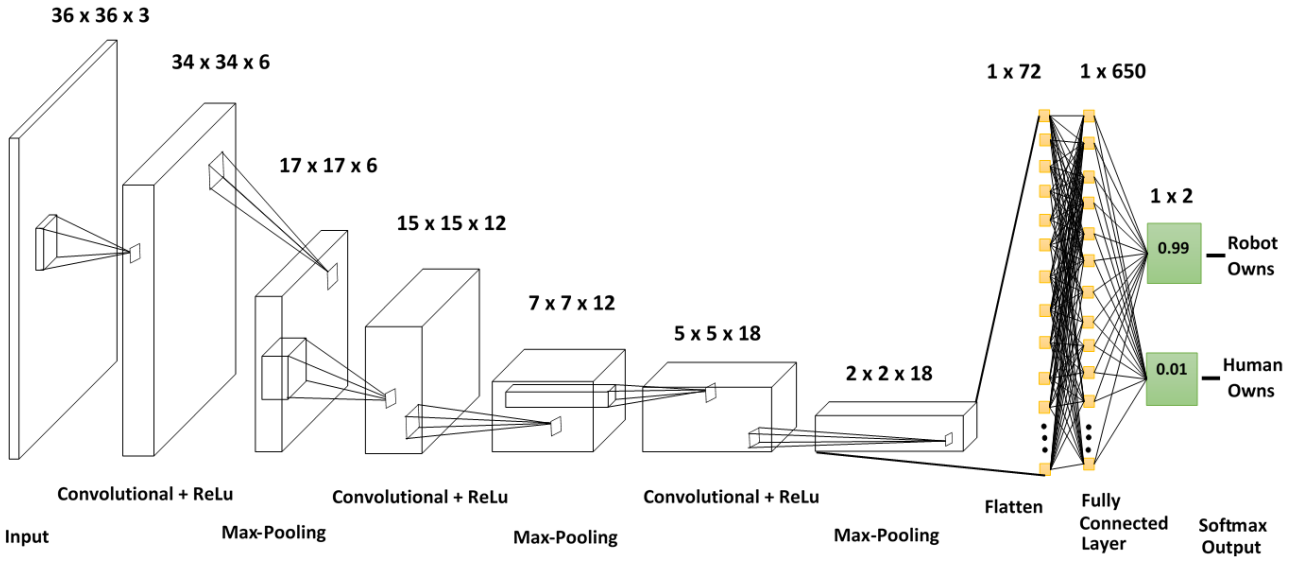


Figure 3.11. Example CNN Architecture of the proposed 36x36-6f OWNET model

Architecture	**Models**							
	1		**2**		**3**		**4**	
Layer type	**36x36-6f**		**36x36-30f**		**72x72-6f**		**72x72-30f**	
	Output Shape	Paramaters	Output Shape	Paramaters	Output Shape	Paramaters	Output Shape	Paramaters
1st Conv Layer with ReLu	34x34x6	168	34x34x30	840	70x70x6	168	70x70x30	840
1st Max pooling layer	17x17x6	-	17x17x30	-	35x35x6	-	35x35x30	-
2nd Conv layer with ReLu	15x15x12	660	15x5x60	16,260	33x33x12	660	33x33x60	16260
2nd Max pooling layer	7x7x12	-	7x7x60	-	16x16x12	-	16x16x60	-
3rd Conv layer with ReLu	5x5x18	1962	5x5x90	48,690	14x14x18	1962	14x14x90	48,690
3rd Max pooling layer	3x3x18	-	3x3x90	-	7x7x18	-	7x7x90	-
Flattened layer	1x72	-	1x360	-	1x882	-	1x4410	-
Fully connected layer	1x650	47,450	1x650	234,650	1x650	573,950	1x650	2,867,150
Output layer	1x2	1302	1x2	1302	1x2	1302	1x2	1302
Total trainable parameters	**51,542**		**301,742**		**578,042**		**2,934,242**	

Table 3.2: Architectures and number of trainable parameters for each of the OWNET models

3.3.3 Overfitting

Over-fitting is an error that can occur when training a model where the function is fitted so closely to all the data points that it cannot generalise to data it has not been trained on.

When training with high numbers of parameters, it becomes much easier to over-fit the model (Muller and Guido, 2017). This is characterised by the training error continuing to trend towards 0 whilst the validation error begins to increase. In this case early-stopping should capture the data at a local minimum, this can be seen in Figure 3.12.

Two methods for handling the problem of over-fitting have been used in this model; one involves manipulations to the data, known as 'data augmentation', and the other, a technique known as 'dropout'.

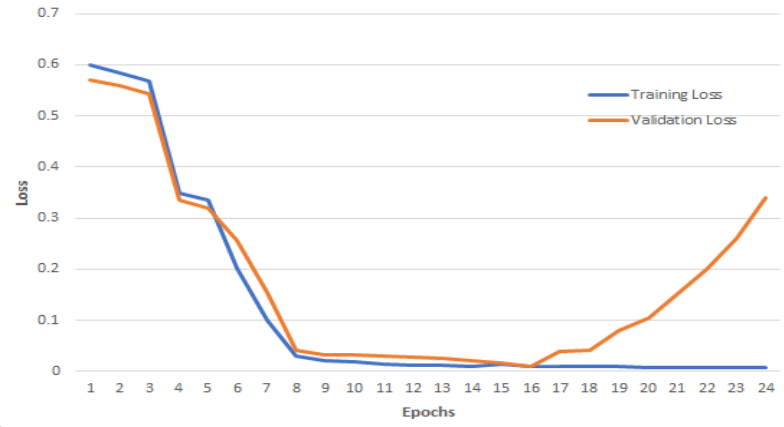


Figure 3.12. Example of an Over-fitted model

3.3.3.1 Data Augmentation

Augmentation is a well documented method of reducing over-fitting, it is the process of artificially modifying images of your current dataset to make it larger (Wong et al., 2016; Perez and Wang, 2017). These transformations must preserve the labels on the images, the goal is to artificially

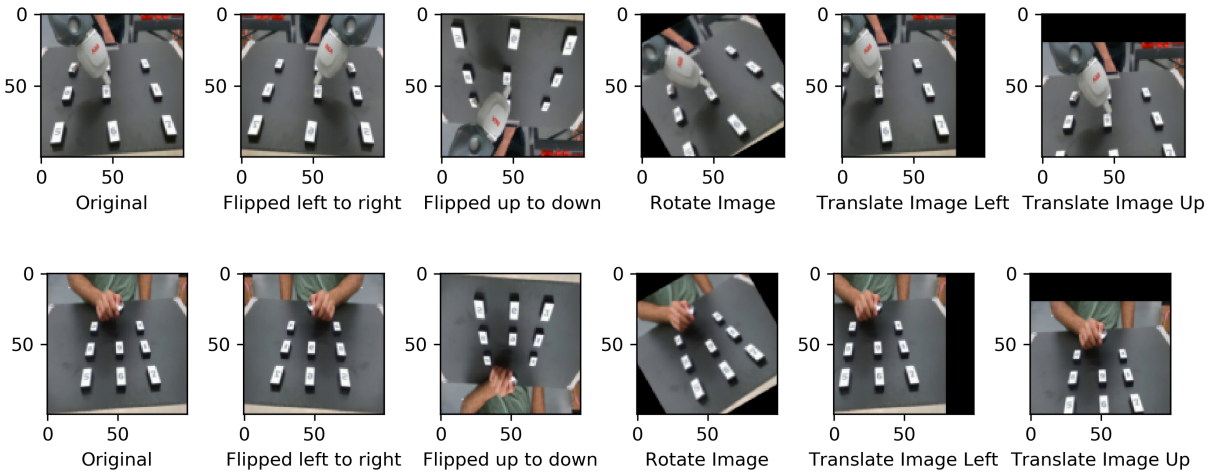


Figure 3.13. Examples of augmented training data, labelled with the augmentation style applied

alter the data in a way that will enhance the networks generalisability, not just in reducing over-fitting. Keras provides a class for doing this, called Image_Data_Generator, which uses the CPU to complete chosen augmentations randomly whilst the GPU trains the previous epoch of the model, so it is computationally inexpensive. For this thesis, five augmentation options were used, these included: flipping the image horizontally and vertically, image translations in

the x-axis and the y-axis, and finally image rotations. Examples of both user and robot labelled original images and their augmented versions can be seen in Figure 3.13.

These were picked by the author as it was felt that they best represented possible similar images that may be needing classification. Different orientations, translations and reflections represent possible images that could have been collected given the time.

Alternatives which were not used in this thesis included altering intensities of the RGB pixels, inverting, gaussian noise, zooming in/out and random cropping.

3.3.3.2 Dropout

One method to increase accuracy in testing scores, which has been very successfully applied, is to combine several different models to the same problem and take the average of all predictions (Bell and Koren, 2007); this method works well but does not scale well with models that take a lot of time to train and suits models with larger numbers of classes. It is also a rather time consuming task tuning hyper-parameters for multiple architectures. Dropout, addresses the issue of over-fitting and trains one network as if it is multiple networks, aiding accuracy in testing conditions.

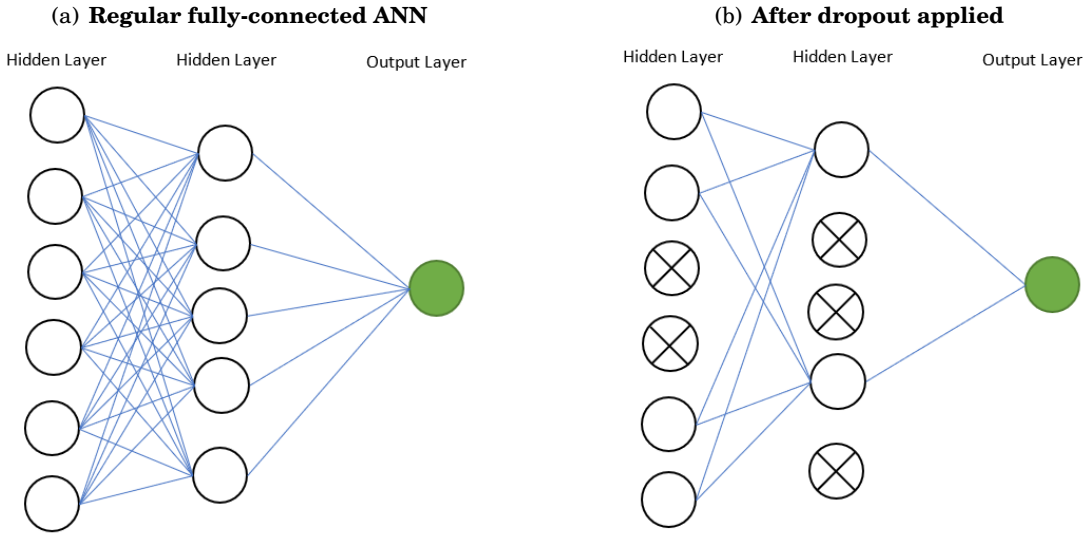


Figure 3.14. Diagram of Dropout concept; the figures above show a regular fully connected feed-forward neural network (a) and the same network after the Dropout technique has been applied (b)

Dropout is a method which, like its name suggests, drops certain neurons outputs in the layer it is used in, effectively making them zero (Srivastava et al., 2014). This is not a permanent feature of the network, it occurs with probability $\boldsymbol{p}$ in each epoch of training, it resembles sampling techniques. Neurons that survive and are not 'dropped' still contribute to the forward-pass and back-propagation calculations. Therefore, with each iteration a new architecture is effectively

used, and these architectures will share weights as normal. This prevents co-adaptation where neurons rely on other neurons to be present, forcing more reliable features to be learned. Dropout was applied originally to the fully connected layers in CNNs at ***p = 0.5*** (Srivastava et al., 2014). It has since been utilised in the convolutional layers too, with ***p = 0.2*** (Graham, 2014).

In this thesis the model applies both of these dropout methods to the network. In the convolutional layers, neurons have a probability of dropping out with ***p = 0.1*** and the fully connected layer has a probability of ***p = 0.5***. This does cause an added cost as it was found that convergence to a global minimum is increased by a third.

3.3.4 Model Learning

The models softmax outputs were trained using the 'Adam' optimisation algorithm, which Keras provides, and is adapted from the original paper proposing it (Kingma and Ba, 2014). This optimises the categorical cross-entropy loss between the two output nodes on a batch of 30 input examples. The parameters used for the algorithm were a learning rate of 0.01, weight decay was set to 0.0005 as in Krizhevsky et al. (2012), beta1 was 0.9, beta2 was 0.99 and epsilon was 0.1. The categorical cross-entropy loss measures the performance of a model where the output is a probability between 0 and 1, it increase when the prediction diverges from the label, it is calculated on all of the output classes and then summed for the total loss. See equation 3.3 for a mathematical description of the loss calculation, where M is the number of classes, log is the natural log, y is the binary indication of whether classification **c** is the correct given observation **o** and **p** is the probability calculated that observation **o** belongs to class **c**.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{3.3}$$

All weights were initialised using Keras's default method, *glorot_normal*, which draws from a truncated normal distribution centered around a zero mean, where the standard deviation is equation 3.4.

$$\mathrm{Var}(W) = \sqrt{\frac{2}{n_{\mathrm{in}} + n_{\mathrm{out}}}} \tag{3.4}$$

Biases were initialised at 0.1 in all layers containing ReLu activations and 0 for the final activation in the output layer. Early stopping parameters were monitoring the validation loss function, and was programmed to end training when no improvement in the model is recorded. This is equal to when the change in the loss is less than a set minimum delta value, which in this case was a 0.001 decrease; this checking mechanism is designed to prevent over-fitting through unnecessary training once a model has reached its global optimum.

The network has been tweaked through trial and error to arrive at these values. Around 50 training attempts were made to arrive to this architecture. The average training time was

roughly 5 and 10 mins per epoch, for both resolutions respectively, when cycling through a dataset of 39,012 images.

3.4 Ownet-Object

Architecture	**Model**	
	5	
Layer type	**182x182-20f**	
	Output Shape	Paramaters
1st Conv Layer with ReLu	178x178x20	1,520
1st Max pooling layer	89x89x20	-
2nd Conv layer with ReLu	87x87x40	7,200
2nd Max pooling layer	43x43x40	-
3rd Conv layer with ReLu	41x41x60	21,600
3rd Max pooling layer	20x20x60	-
4th Conv layer with ReLu	18x18x80	43,200
4th Max pooling layer	9x9x80	-
Flattened layer	1x6,480	-
Fully connected layer	1x650	421,200
Output Layer	1x18	11,700
Total trainable parameters	**506,420**	

Table 3.3: OWNET-Object architecture and trainable parameters

Following the previous sections description of the OWNET model. This section will briefly describe a further model and architecture which will be trained using the same learning parameters with a much larger set of classes, 18 in total. One for each of the nine blocks, within a hierarchy of the two ownership conditions from the OWNET model. Data has been labeled accordingly to suit this classification problem previously, with each image following extraction being labeled with not only the ownership but also the brick being owned (e.g "R-1", "R-2, "H-1", "H-2").

For the OWNET-Object, 182x182 size resolution was used for input images as numbers on blocks could be determined at this resolution by the human eye. The network built according to the same architecture as OWNET with the addition of one extra convolutional and pooling at the beginning with a kernel filter size of 5x5; see table 3.4 to see a summary of the architectures layers. The model was trained using the same hyper parameters as the previous OWNET model, described in section 3.3. If this model is able to produce reliable classification results this will be highly impressive as the distinguishing features represent such a small amount of the input image.

Conclusion

This chapter has covered a brief introduction to machine learning methods and centred on a neural network methodology to be used to build a model of ownership classification. The particular branch of neural networks which will be used to train this model is convolutional neural networks (CNN) which have a strong architectural link to the human visual system which the author suggest must be important to RFT principles.

An overview of the theory behind CNNs has been explained and their use justified. A section on the data collection process followed this, where the entire procedure was explained, including

relevant information on the participants and equipment used. The final two sections covered the proposed model architectures, hyper-parameters and learning algorithms to be used with the dataset. These two models are composed of a high level problem and a low level problem; classifying ownership in a situation and classifying the ownership and the object of interest. This will be used as an inference system in human robot interactions to enhance user experience.

The code used for compiling, training and statistically evaluating the models can be seen in appendix E.

The following chapter will describe the results gained from the training and testing the OWNET models and the OWNET-Object model.

CHAPTER 4

RESULTS

The following chapter contains three sections, 4.1 defines and explains the metrics used to evaluate the models which have been trained. This is followed by 4.2 which describing and evaluating the training and testing metrics for the OWNET model. In total there were four models of OWNET trained and tested, as described in chapter 3, the independent hyper-parameters being compared are the input image size and the density of the feature maps. A model will be selected, based on the best metrics score, for qualitative analysis. The final section of the results, 4.3, describes and evaluates the trained OWNET-Object model.

4.1 Metrics definitions

During the 4.2 there will be four main metrics used: F_1 score, recall/ true positive rate (**TPR**), precision and training/ validation loss graphs. A subjective measure, qualitative analysis, is also used on the best trained OWNET model. For the OWNET-Object model top-k error rates were used to measure the models ability.

The first three are measures are binary classification metrics; as a softmax function was used in the output, the output predictions are a percentage of whether the image belongs to that class, so these were converted to binary counts: true positive (**TP**), true negative (**TN**), false positive (**FP**), and false negative (**FN**). For this thesis, when the prediction in the correct class is above 0.95 (95%) a TP reading is recorded, if less then a FN is recorded. A FP is recorded if the opposite class is predicted with more than 0.5 (50%), if less then a TN is recorded. It could have been possible to set the thresholds both at 0.5, however, to really test the model, the TP count was made more strict. These metrics were calculated using the program 'stats.py' (Appendix I). The graphs showing training/ validation loss over time were constructed with the outputs loss function from each epoch in the models training process.

The TPR or recall can be defined as the ratio of correct classifications over all the possible correct instances, the precision is the ratio of correct classifications over all classifications made. Both look at relevance of the classification made, however a precision ratio alone would just show that all classifications made were correct, not what was missed; a perfect TPR would show all correct classifications were made, but not what incorrect ones were classified.

An F_1 score is a binary statistical measure and a robust indicator of the classification accuracy as a model, it also known as the harmonic-mean between recall and precision according to Powers (2011). A high F_1 value indicates a more robust classification model, at its best is 1 and worst 0.

These equations are described mathematically in Equations 4.2 - 4.3; where **TP** is the true positive count, **FP** is the the false positive count, **TN** is the true negative count, and **FN** is the false negative count.

$$\text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.2}$$

$$\text{F}_1 = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{4.3}$$

A basic qualitative analysis was completed on a set of images which produced both correct and incorrect predictions. This involves commenting on possible features which may have influenced the classification scores. This is a largely subjective measure, but can be useful for understanding what the model has considered important during training.

Top-k error is a metric for multi-class classifiers and shows the ratio of examples not predicted into the top-k prediction classes; the k is variable but typically the top 1 and 5 prediction classes are used. For example, if there were 100 classes and the correct classification was in the top 5 prediction percentages, it would be a member of the top 5 for that iteration. The top-k essentially calculates the ratio where this doesn't occur, so the smaller the figure the better, as that means it has a low error rate.

4.2 OWNET

This section describes the results from the four OWNET models, with some assessments and evaluations of what the data shows us about these models.

4.2.1 Training and validation performance metrics

Figures 4.1 and 4.2 show the hyper-parameter, number of epochs trained, on the x-axis and the loss/ error function on the y-axis, signalling the networks quality as training persists. This graph is useful for gauging whether a model has over-fit or not.

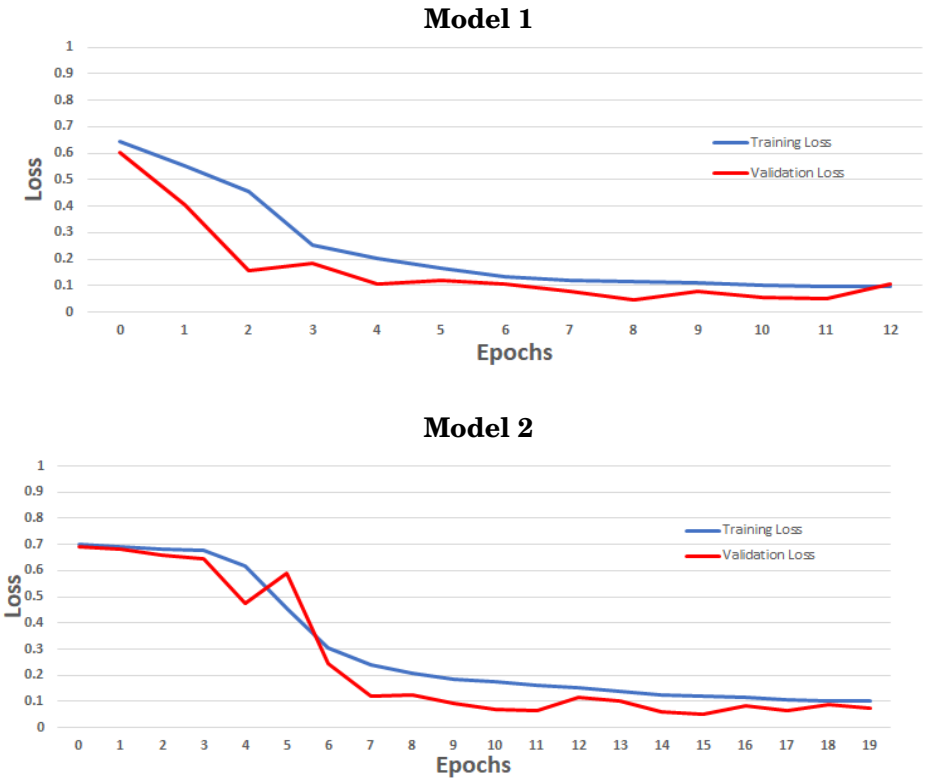


Figure 4.1. Loss graphs of OWNET models 1 and 2; model 1 contains 36x36x3 input and an initial feature map size of 6, whilst model 2 contains 36x36x3 input and an initial feature map size of 30

Comparing the two models in 4.1 it can be seen that model 1 begins to decrease from the first epoch whereas the model 2 plateaus at the start before the loss reduces rapidly. Model 1 also appears to show signs that it was over-fitting in epoch 12 as the loss begins to increase, which will have triggered the early stopping parameters for the model. The plateau for model 2 could indicate that the learning rate was not appropriate for the amount of trainable parameters in the network, however most likely this is due to random training differences in the initialisation of the weights.

In comparison, model 3 and model 4 in 4.2 appear to have trained appropriately, with a slightly steeper reduction in the loss after the initial epoch in model 4. All models training and validation loss, apart from model 1 at the last epoch, show lower validation loss than training. The final loss function figures for Model 1 - 4 in order can be seen in table 4.1 below.

Whilst all models manage validation losses below 0.1, interestingly, model 1 shows the best

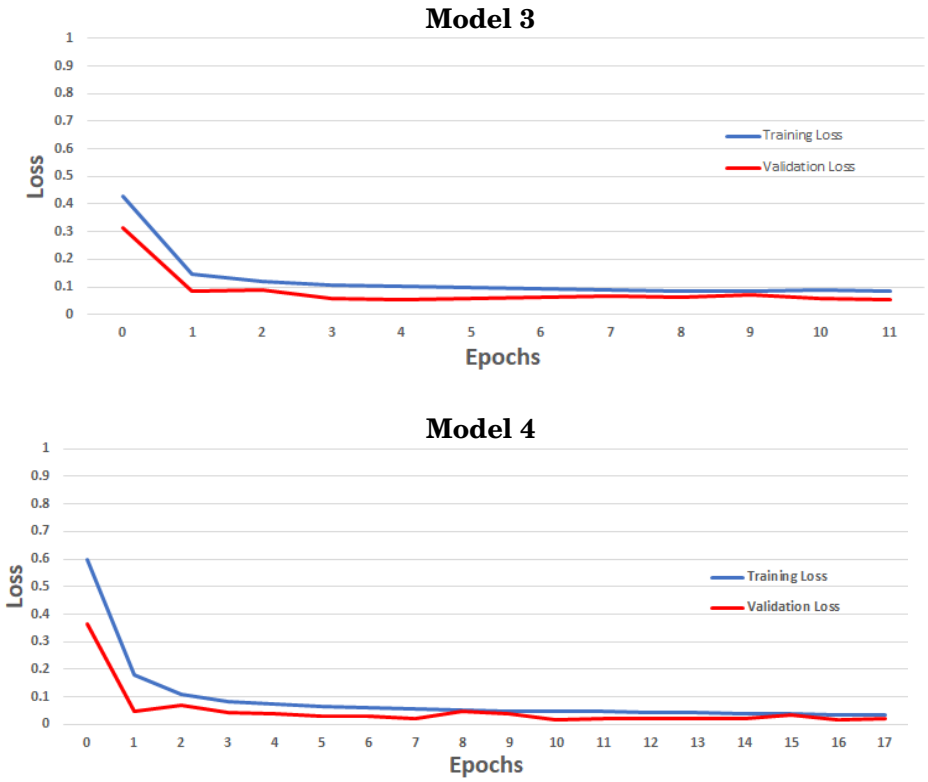


Figure 4.2. Loss graphs of OWNET models 3 and 4; model 3 contains 72x72x3 input and an initial feature map size of 6, whilst model 4 contains 72x72x3 input and an initial feature map size of 30

validation loss at the final epoch. Model 4 has the lowest when both training and validation are considered from both the graphs in 4.1 and 4.2 and Table 4.1. It is not clear which initial feature map size works best, the resolution however, appears to produce the strongest results when the image input resolution is at 72x72.

Model		**Final Loss Function Values**	
		Training	**Validation**
1	36x36-6F	0.09802	0.01059
2	36x36-30F	0.10281	0.07606
3	72x72-6F	0.08469	0.05556
4	72x72-30F	0.03497	0.02035

Table 4.1: Final loss function values from each of the four OWNET models (1-4) at their final epochs

4.2.2 Testing set performance

All models were tested on the same 4,935 images from two of the data collection trials, numbers 3 and 9. Table 4.2 below, shows the recall, precision and F_1 scores for each of the models.

Model		Performance Measures		
		Recall (TPR)	**Precision**	**F_1 Score**
1	36x36-6F	0.620704	0.961513	0.754404
2	36x36-30F	0.689441	0.884227	0.774779
3	72x72-6F	0.812422	0.866225	0.838462
4	72x72-30F	0.850932	0.925676	0.886731

Table 4.2: Performance measures for OWNET models 1-4; including the recall, precision and F_1 score

The highest performing model according to the F_1 score is model 4 at 0.89; this score declines in both resolution conditions with less feature maps, with variances between model 2 and 1 of -2.7% and model 4 and 3 -5.8%. There is also a difference between the two resolutions, with model 3 and 1 having a variance of -11.1%, and model 4 and 2 at -14.4%. It appears that when the resolution is lower, the models are more precise, gaining less false positives, but the recall rate is sacrificed with more false negatives appearing. This is probably because it is harder for the network, for example if one of the feature map happens to encode the edges of a hand and the gripper; the similarities in the shapes at lower resolutions would be greater, resulting in greater uncertainty between the classifications, therefore the 95% threshold is unlikely to be met to classify this as a true positive.

4.2.3 Qualitative Analysis

Model 4 has been chosen by the author, to undergo a qualitative analysis, as it represents the most robust model based on having the highest F_1. The two Figures (4.3 and 4.4) in this subsection show examples of robot owning conditions and human owning conditions; each show six images which have been subjectively labelled roughly by the percentage prediction of the correct class.

These labels include: 'Poor', representing a lower than 20% prediction; 'Undecided', where the model is split between both classes, roughly when the prediction is 40-60%; and 'Good', when the prediction is greater than 90%. Images have all been lowered to the resolution entering the trained network (72x72).

When comparing the images in 4.3, **c** and **f** both have the robot arm orientated to a position where the smart grippers sensor is in view of the camera, which at this resolution, shows as a dark patch in the centre of the gripper; images **a**, **d** and **e** all do not have this, and **b** is showing only slightly. It may be that this feature the network has considered useful in determining between the two conditions, if a feature maps convolutions picked out edges, the shapes between hand and gripper may be similar at this resolution, and having the dark patch may set them apart.

Also turning to image **d** in 4.3, it is possible this returned a poor prediction as the features do not show as well with the gray background of the floor behind the gripper. Equally, if the feature being selected for is the colour, there appears to be more skin appearing in pixels in the poor and

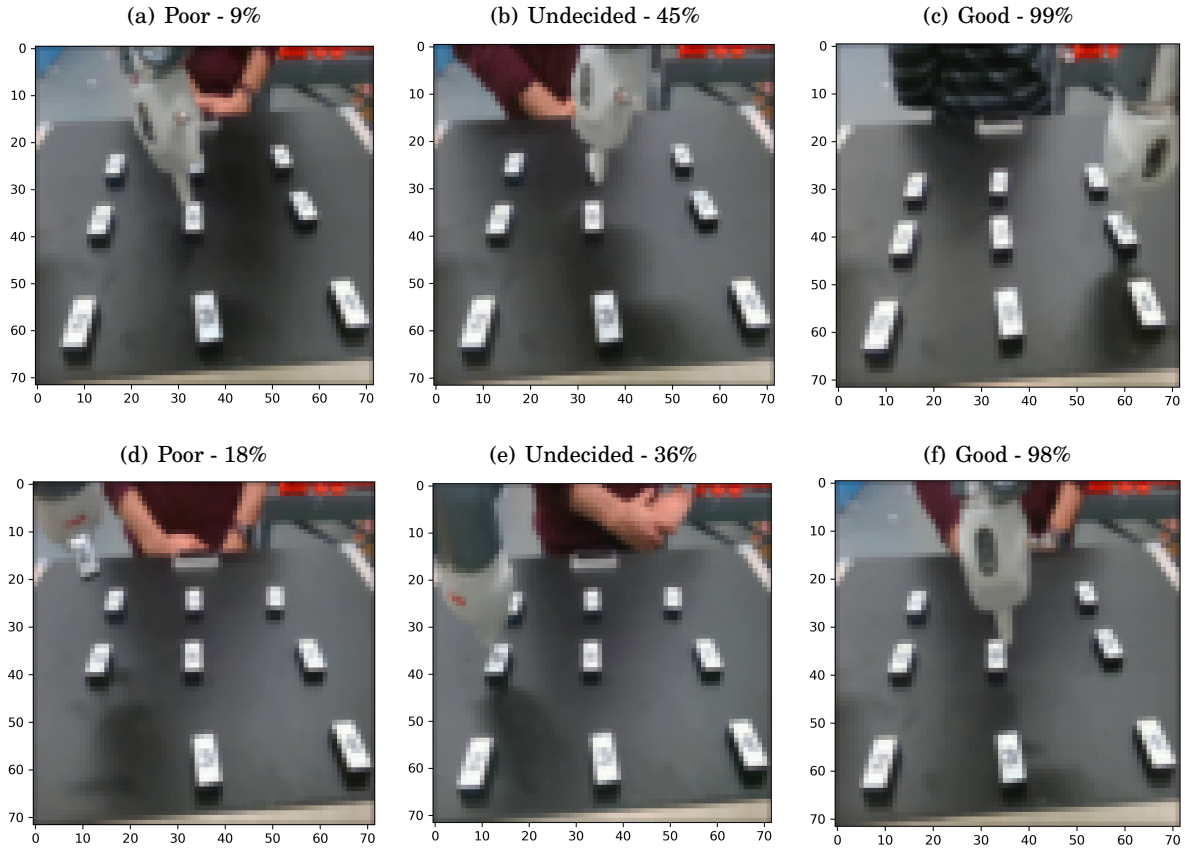


Figure 4.3: Qualitative analysis data for robot ownership classifications, the images shown here represent three categories; poor, undecided and good classifications

undecided images, whilst the good images have less.

In Figure 4.4, the differences in prediction accuracy for human ownership is less clear. For example, none of the images contain pieces of the robot arm or gripper, as opposed to some skin pixels appearing in almost all examples in Figure 4.3. It could be that the colour of the board and floor, which is similar to the robot arm, activate enough of the neurons in the feature maps that encode colour. So in the absence of other features, such as the shape of the hand in **d**, the network resorts to other learned features, such as colour to make the prediction[1]. The pose of the finger in **b** could be resulting in poor predictive ability in that image as typically that pointing pose was not made, but the class may still achieves partial activation due to other features. This is also the case in **e**, where part of the hand is out of view of the camera, essentially cropping half of the hands features.

[1]Image **d** in Figure 4.4 was an error in labelling, as no party 'owns' a brick in this image. This image on further inspection was taken 32 frames into recording, during processing as images were being gathered from videos the initial 30 frames and last 30 frames were cropped from selection due to this error in labelling.

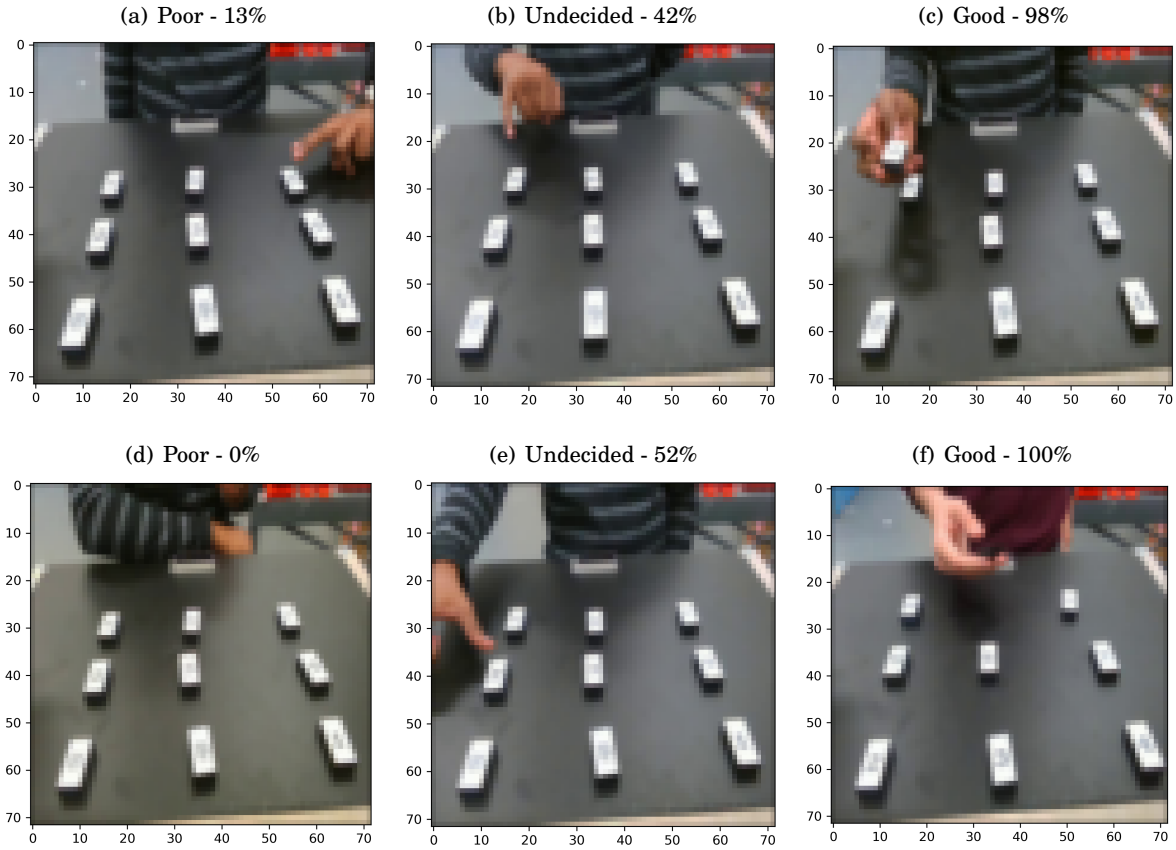


Figure 4.4: Qualitative analysis data for human ownership classifications, the images shown here represent three categories; poor, undecided and good classifications

4.3 OWNET-Object Classification

As discussed in chapter 3, a model which was designed to classify not only ownership but the object being owned was trained. In this section the results of that training will be described and discussed. Using slightly different metrics which are more appropriate with such a high amount of classes; top-k error rates, discussed 4.1 were used, and discussed in 4.3.2. Like in 4.2, a validation and training loss graph with epochs along the x-axis will be described and shown in 4.3.1.

4.3.1 Training Metrics

As can be seen in Figure 4.5 the training plateaued immediately and the loss in both the training and validation sets failed to decrease. This was the case for all attempts at training this model, multiple parameters were used in both the architecture (altering the number of layers) and the optimisation algorithm (tuning the learning rate).

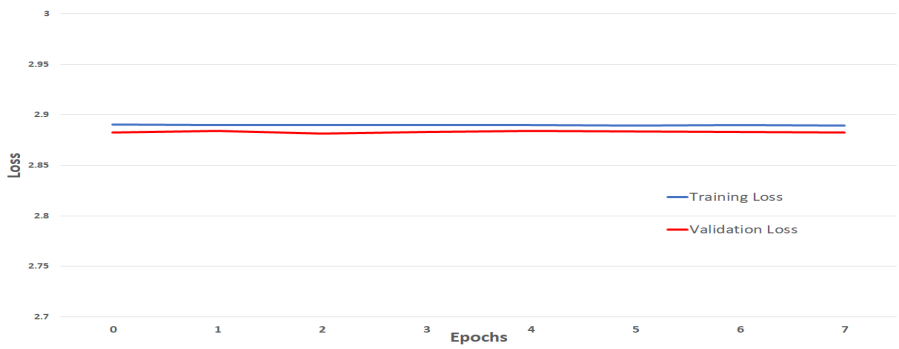


Figure 4.5. Loss graph in the training and validation sets after each training epoch with input images of 182x182x3 and initial feature map size of 20

4.3.2 Testing Scores

The testing scores seen below in Table 4.3 show the error rates for the model. With correct predictions being the top prediction score only 4.3% of the time; in the top three 14.9% of the time and appearing in the top 5 36.2% of the time. These figures show that the model was no better than chance at predicting the correct class.

Model		**Top-K error ratios**		
		Top-1	**Top-3**	**Top-5**
5	182x182-20f	0.95744	0.85106	0.63829

Table 4.3: Top-K error ratios for the trained OWNET-Object model

This is an unsurprising set of results given the validation and training loss graph. On closer inspection the model was predicting the same set of output predictions for every one of the 5,000 test images. This will be discussed in the discussion section further.

Conclusion

The results described in this section represent promising findings for the thesis in the case OWNET. Out of the four models trained and presented in 4.2, the best classifier was model 4 which had an input size of 72x72x3 and an initial feature map of 30. The OWNET-Object model, however, reported on in 4.3 completely failed to classify objects or ownership with any degree of accuracy. The results from this chapter and methods from chapter 3 will be evaluated and critically appraised in the discussion in the next chapter.

CHAPTER 5

DISCUSSION

Following the previous chapter which covering the results from OWNET and OWNET-Object. this chapter discusses these findings, their potential benefits, applications and limitations.

Section 5.1 evaluates the method used in producing OWNET and OWNET-Object, based on the results obtained in the previous chapter and in the context of the aims of this thesis. 5.2 discusses an example of how it could be used in potential human-robot interaction situations. This also covers the ethical implications of using the technology in HRI situations.

5.3 critically appraises the limitations of this thesis at tackling the problem of classifying ownership as a precursor to perspective taking. Finally, 5.4 suggests possible ways to enhance the results in the future and other methods that could be utilised in future works in the theme of relational frame theory.

5.1 Results discussion

This section contains two subsections, 5.1.1 and refeval:OWNET-Object. The former covers the evaluation of the OWNET models, whilst the latter focuses on evaluating the OWNET-Object.

5.1.1 OWNET method

The aims discussed in this thesis were to use a machine learning method to learn the precursor, to perspective taking, ownership. Following research of the methodology, convolutional neural networks (CNN) architecture was chosen as the best method for learning the relationships from image data. The results from the OWNET models show that a relatively large CNN can classify situations of ownership in interactions with a robot. With model 4 being the most successful

at delivering this prediction of ownership. The predictions exhibit a strong balance between precision and recall, in other terms the model, lowers false positives whilst still ensuring it is correctly classifying a large ratio of the possible correct classifications to be made.

Whilst using the high-level API Keras, it is not possible to easily view the kernel convolutions in each layer when an image passes forward through the network. Ideally, visualising these convolutions, would allow for better interpretation of what features the model finds useful at making these classifications. As each layer abstracts from the image more it becomes harder to predict what the model is using to correctly interpret the condition. As each feature map in each layer contains a different kernel filter with some set weights, it becomes difficult to understand what might be activating different receptive fields within the network.

It is most likely, in the authors estimation, that edges of the shapes between hand and gripper play a large part in these features; it is also highly likely following the qualitative analysis that colour of the gripper against the skin colour of participants was a distinctive feature for the model to fixate on.

This presents one of the largest problems when working with neural networks, their inner workings are best known as a black box (Siciliano and Khatib, 2016). They are just function fitting devices, with their optimisation algorithms just attempting to fit input to output. Given enough data, computational power and the right model, they should be able to be trained to fit any function as they are universal approximators (Hornik, 1991).

Overall this model was successful in the aims of this thesis, it very accurately performed classification of ownership on test data.

5.1.2 OWNET-Object method

The objective of this model was to not only successfully classify which party had ownership in the image, but to also classify which object the party had ownership of. From the results in 4.3, we can see that this model was worse than chance at classifying objects and ownership together.

This is likely to be the result of the approach and methods used, combining two problems over complicates the model. With a deeper network and much more computational power, it may have been possible to achieve good results using this methodology. It is also possible that the network is constrained by the small amounts of data for each class (2500 example images per class, compared to OWNETs over 20,000 per class). As the object being classified had identifiers, numbers, printed on them it was reasoned that they would be sufficient information for the network to pick up as features. It is likely that the features represented too small of a space in the network to make a large enough difference to the output node predictions. In the absence of other differences in the images, the network stagnated, not being able to make alterations in the weights of kernel filters to reduce the loss error function.

5.2 Applications

According to Feil-Seifer and Mataric (2005), in order to develop socially assistive robotic systems, these systems must engage users and have a perceived level of autonomy. All social robotic situations rely on satisfaction of the user to ensure reuse (Card, 2017), part of ensuring satisfaction should be intuitive ease of use. The author suggests that there is no more intuitive method of communication with an artificial entity than the methods humans use day to day when interacting with other humans. Building an accurate system that can take full advantage of the these qualities would require principles of RFT to be used.

5.2.1 will describe an example of the I-You deictic frame reversal used in a simulated environment from test example interactions with the ABB-YUMI robot. Followed by a number of hypothetical scenarios which the model of classifying ownership could be applied. 5.2.2 will briefly discuss the ethical implications of exploiting biases of users.

5.2.1 OWNET and human-robot interaction examples

A hard-coded set of responses were built onto the end of the classifier, so that when asked by the user one of two questions ("What am I holding"; "What are you holding?"), the classifier makes a prediction with the current input of the hypothetical environment. Based on this prediction for who owns the object in the situation, an if-statement chooses the appropriate response for the agent to make. See Figure 5.1 for an example of this; the code used for this is available in appendix H. The hard coding is a temporary solution in the authors eyes, ideally further research will enable the use of natural language processing to parse more complex inferences based on the input question and ownership prediction. This is discussed further in 5.4.

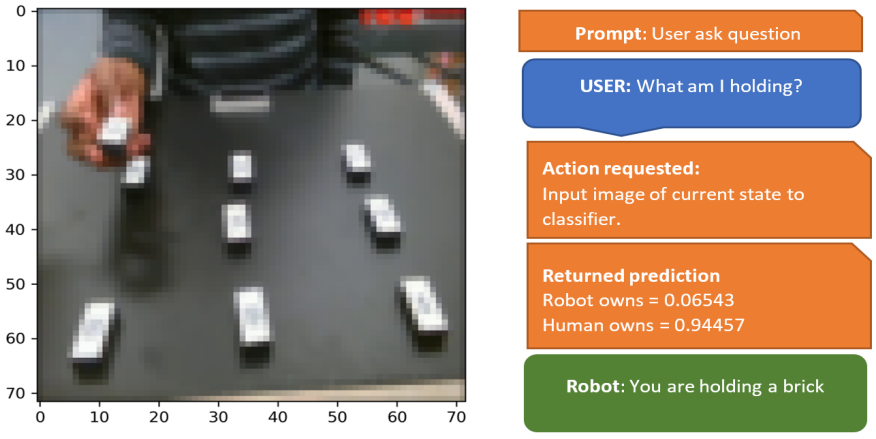


Figure 5.1. Annotated image of how OWNET would be used to return I-You reversal with a user

Examples of more complex situations where recognising ownership, perspectives and ulti-

mately I-you relations, would be assistive robots in settings such as: the factory line; caring for the elderly; tutoring; physical therapy; assisting autistic individuals with emotional expression. Taking one of these examples, physical therapy. One of the main issues that significantly improves outcomes for individuals recovering from conditions such as strokes is repetitive exercise in affected areas (Beattie et al., 2002). Robotic therapists have been employed already to take on the growing number of patients requiring physical therapy (Fasoli et al., 2004). With robotic intervention, the most important tasks is creating an engaging experience for the user to retain the behavior of completing the exercise at the prescribed intervals. When demonstrating exercises to the user, the robotic system could refer to itself as '**I**', "**I** am now going to show **you**..."; when passing to the user, it can also state "**You** should now have a go at...". When the human interacts back, with a query such as, "Can **you** show me that again?", the robot should be able to recognise that the user is referring to the robot and reverse the '**You**' with **I**. The author suggests that by using I-You relations, intelligently, a robot can create a more engaging experience, therefore ensuring patients completion of the therapy.

From the psychology fields point of view, the applications of this technology could be used as a way to test theories in RFT. One issue with psychological research is that it suffers from validity and replicability issues due to the extraneous variables that come with working with human-human experiments (Coolican, 2017). This means that you cannot validate results with high degrees of certainty that what your hypothesis matches the data gained from participants, especially when working with abstract concepts. By controlling one of the participants completely, psychologists could test independent variables involved in RFT to add further evidence behind the theory.

5.2.2 Ethical implications

One ethical problem with enhancing human-robot interaction (HRI) and creating more 'life-like' behaviors, which is the goal in employing RFT principles to this area, is that it plays on the human bias of anthropomorphism. Anthropomorphism and anthropomorphic design is the fallacy of humans to see human like qualities and features in non-human entities (Duffy, 2003).

The evidence of human's ability to anthropomorphise devices and objects is strong (Epley et al., 2007), even with entities totally devoid of human-like characteristics, such as bomb disposal robots ("Funerals for Fallen Robots", Garber, 2013). Even toys have sparked relationships with their user's, Tamagotchi's were artificial pets, built into a small simplistic device that users had to care for, often sparking months if not years of interactions. There are some benefits to playing on this bias, for example, evidence suggests that it increases the facilitation of HRI, by creating familiarity with the system and building on established human-human interaction protocol (Syrdal et al., 2008).

The disadvantages of using this bias will largely fall into two categories, negative ethical implication and where the use is a detriment to the interaction. For the latter, an example would

be in medical check-up circumstances, which can be personal, Bartneck et al. (2010) found that patients preferred less human-like robots. Ethical reasons which could be disadvantages include, the loss in human contact, personal liberty, infantilizationof the user; even concerns that those with certain conditions would choose to take care of a robotic figure even at the price of their own health (Sharkey and Sharkey, 2012). So, it is important that in the pursuit of more realistic technology that the benefits and downsides are considered together in a holistic manner.

When employing methods in artificial agents to control behaviour of users, the designer must play a key role in acting ethically. Especially in situations where users are in a position of reduced mental faculties, such as when physical therapy robots work with stroke patients.

5.3 Limitations

There were two main limitations to the current work completed in this thesis; the dataset (5.3.1) and the theory (5.3.2). The next two subsections will go into details on these two limitations before offering the possible directions of future work in 5.4.

5.3.1 Dataset

The dataset represents two classes, robot owns and human owns, for the OWNET models. In the OWNET-Object model it represents 18 classes, the ownership and the brick being owned. In total 45,000 labelled example images were used from the 18 interactions with the robot and participants described in chapter 3. Theoretically, when planning this thesis, it was considered that 30 frames could be taken per second from footage of interactions, equating to roughly 432,000 examples to train the CNN with. In reality, the differences between each image when being used frame by frame was probably not large enough; this resulted in early models becoming over-fitted. In the end, it was decided that frames would be taken at intervals of 8, limiting the size of the dataset to 48,600, therefore measures were taken to artificially increase the dataset size as a result of this.

During the planning of the thesis, it was considered appropriate to constrain the environment in order to focus on the model being able to learn the principle of RFT; this had the limiting effect of possible generalisability to other situations, which is usually the goal with the use of neural network models. This constraint was probably unnecessary in reflection.

As the models were trained on whole images to classify a situation of ownership, object detection was not considered a goal in the project, therefore regions of interest were not labelled. In future works, it may be useful to label the data with regions of interest in order to perform deeper analysis on those areas. Finally, on reflection, blocks of different sizes, shapes and colours would have been used to introduce greater variability in the dataset; the camera would have been mounted at different perspectives in order to also introduce greater variability.

5.3.2 Theory and methodology

As discussed during the literature review in Chapter 2, relational frame theory (RFT) is thought to be the reason that humans have developed language. One of the primary ways humans interpret the world is through our visual system, if RFT is correct then one of its mechanisms must be the visual system. Therefore a convolutional neural network (CNN), with its basis being the pathways from ganglion cells in the retina to the visual cortex, was deemed the perfect methodology to use to model this theory.

Despite the similarities shared by CNNs and the visual system, CNNs are still an input output fitted function. To the extent that the model has learned a perspective taking ability such as ownership is limited only by the data that the model was fitted to and the architecture of the model. Deictic frames, and more importantly RFT in general is a concept of human learning, it is fundamentally an abstract explanation in itself; it is possible that the most important use for robotics will come from emulating the functionality rather than actually attempting to simulate the concept in an artificial agent.

This exploratory piece of work looked at whether situations of ownership could be classified. There is most certainly a debate to be had over whether the OWNET model has learned ownership prediction. The results suggest that they can, but more work is needed to develop a model that has more generic abilities in this classification goal, in order to be useful in wider HRI systems. The question of whether the model represents a true reflection of RFT may be irrelevant if just replicating the behaviours of the theory gives the main benefits.

5.4 Directions of future research

This section describes, based on this discussion, the directions the author would like future research into this topic to go in. 5.4.1 covers the improvements which could be made to the dataset following this thesis. Following this 5.4.2 discusses different methods which could be useful to explore in emulating RFT in more depth. Finally, 5.4.3 recommends that other modalities, other than image data, be looked into for incorporation into further models in the future.

5.4.1 Dataset Improvements

As with most machine learning problems, more data is key. The dataset could be improved in any future work, by building on different objects, colours and perspectives of the camera position. This should enable greater variation between the images. Moreover the labels can be improved, by adding regions of interest (ROI) co-ordinates which could be used instead of general classifications on the whole image. This could enable the network to select a ROI and perform a further task on the selected region. These ROI could be the hand or gripper. This would require more human hours to go through and label the ROI in each image.

5.4.2 Methodology of classification

CNNs are most likely the best machine learning method for parsing image data due to their superior processing time and abilities in abstracting features (Krizhevsky et al., 2012). However, the image as a whole contains information which does not aid classification, which in the OWNET-Object model likely became an issue. One method which could be explored to get around this is for a model to be trained on localising the hand or gripper to a ROI, as discussed in further labelling to the dataset. Using this information to crop the image and perform an object detection model which solely focuses on classifying objects within it, such as the number on the block being manipulated.

As for language processing, much of the decisions in how humans respond to situations is not informed from a single image alone, but a series of images. It would be interesting to apply long short term memory (LSTM) networks with embedded CNNs to this problem. LSTM networks have an architecture which allows them to take into account the series of information which has just passed through in the previous iteration, which informs the predictions and classifications they make in the present iteration. LSTMs have been used with CNNs to parse through sets of sequential images series and apply captions to the images based on what the network concludes is in the image (Vinyals et al., 2015). This technique could be applied to modelling RFT, by introducing vectorised sentences to an LSTM network with an embedded version of OWNET to output an appropriate output response based on relational frame principles. One method which appears to have been successful, in test conditions, was the research completed by Santoro et al. (2017), where the relational module was introduced to network architecture; this module was able to make inferences about objects in an image based on their relationship to other objects in the image. This method was not similar to the theory or problem behind the current thesis, but this could be looked at in more detail to inform further research into modeling RFT principles.

5.4.3 Explore multi-modal information

Another area which could be expanded on is collection of data in other modalities such as gesture, emotion, speech and touch data. When looking into other frames involved with RFT, it would be useful to model systems using information from all of the human senses. This would represent a more holistic life-like model, which would likely be able to make stronger inferences to be used in HRI.

Conclusion

This concludes the discussion on the methods and results of this thesis and its recommendations to future research. To recap, this chapter has focused on a critical appraisal of the methods used in both the OWNET and OWNET-Object models. This chapter also discussed a subset of applications, in which applying RFT principles to interactions could be beneficial, with considerations of the ethical challenges this poses. The author's views of the limitations inherent in

the thesis's methodology was considered and discussed; which lead to some thoughts on possible future research directions for this research theme.

The final chapter will be a summary of the contents of the entire thesis, along with a brief concluding paragraph.

CHAPTER 6

CONCLUSION

This thesis has taken an exploratory approach to using a psychological theory of language acquisition, Relational Frame Theory, and using it to enhance robot interaction. The thesis, completed a review of the theory and its background, before focusing on the Deictic frames, frames which describe a form of perspective taking (Jackson and Hooper, 2017. The aim was to the model perspective taking in an artificial agent, an important aspect of perspective taking, in the author's estimation, is recognising position of ownership of stimuli and events, and the stimuli and events themselves.

The thesis then explored machine learning methodologies which could be used to model this ability and behaviour. Convolutional neural networks (CNN) were ultimately used to tackle this classification problem, and OWNET a deep CNN architecture was proposed. This model needed information in order to learn this ability to classify ownership, so a period of data-collection took place.

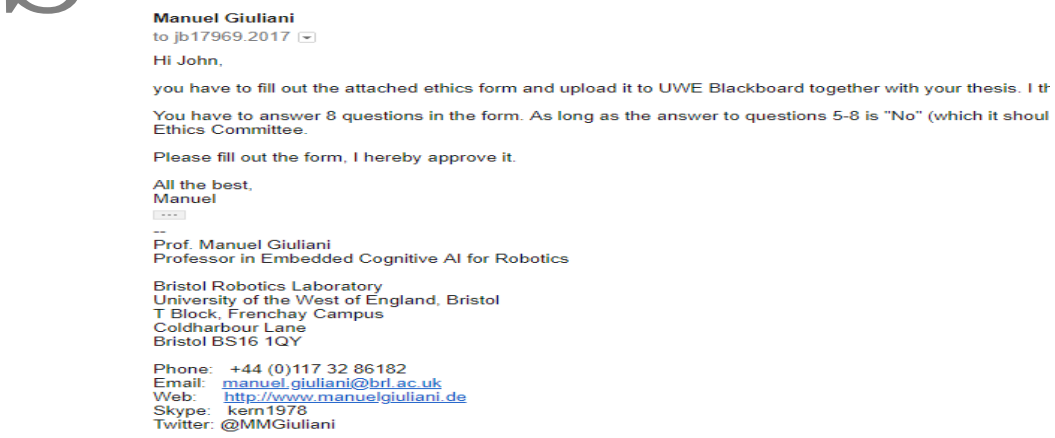
Two main models were finally produced; OWNET and OWNET-Object. OWNET was successful in classifying ownership, particularly when more feature maps and higher resolution was used. This provided the network with more trainable data points to make inferences from for the classification. OWNET-Object was unable to perform ownership classification with object-detection according to the findings of the research. It was discussed that this may have been due to the limited dataset for eighteen different classes, as well as possibly the model was unsuitable for this problem.

This was discussed in chapter 5, and suggestions were made to improve this. Applications of the technology were also discussed in this section, with concerns being raised over ethical issues when designing to manipulate human behaviour. The overall results of the thesis were useful in grounding of RFTs potential benefits to autonomous agents, but as this was initial work

into this theme, it does lack other results to compare it to, making evaluation difficult. Further research should explore the recommendations made in chapter 5, in particular looking to extend the dataset and attempt other neural architectures such as LSTMs.

APPENDIX

See below for attachment of the PDF ethical approval document. See also the screen-shot of the approval from the supervisor.



Manuel Giuliani
to jb17969.2017

Hi John,

you have to fill out the attached ethics form and upload it to UWE Blackboard together with your thesis. I th

You have to answer 8 questions in the form. As long as the answer to questions 5-8 is "No" (which it shoul
Ethics Committee.

Please fill out the form, I hereby approve it.

All the best,
Manuel

--
Prof. Manuel Giuliani
Professor in Embedded Cognitive AI for Robotics

Bristol Robotics Laboratory
University of the West of England, Bristol
T Block, Frenchay Campus
Coldharbour Lane
Bristol BS16 1QY

Phone: +44 (0)117 32 86182
Email: manuel.giuliani@brl.ac.uk
Web: http://www.manuelgiuliani.de
Skype: kern1978
Twitter: @MMGiuliani

University of the West of England
BRISTOL

Faculty of Environment & Technology
Faculty Research Ethics Committee (FREC)

Ethical Review Checklist for Undergraduate and Postgraduate Modules

Staff and PG research students must not use this form, but should instead, if appropriate, submit a full application for ethical approval to the Faculty Research Ethics Committee (FREC).

Please provide project details and complete the checklist below.

Project Details:

Module name	Dissertation MSc Robotics
Module code	UFMEDA-60-M
Module leader	Maryam Atoofi
Project Supervisor	Manuel Giuliani
Proposed project title	**Teaching Robots Perspective Taking**

Applicant Details:

Name of Student	John Birkett
Student Number	10022176
Student's email address	J617969 @ bristol.ac.uk

	CHECKLIST QUESTIONS	Yes/No	Explanation
1.	Does the proposed project involve **human tissue, human participants, animals, environmental damage, or the NHS.**	Yes	*If the answer to this is 'No' then no further checks in the list need to be considered.*
2.	Will participants be clearly asked to give consent to take part in the research and informed about how data collected in the research will be used?	Yes	The participants will receive an information sheet explaining what data will be collected, how it will be stored and used. This will provide participants with fully informed consent.
3.	If they choose, can a participant withdraw at any time (prior to a point of "no return" in the use of their data)? Are they told this?	Yes	The participants as part of the informed consent will be informed that they have two weeks from taking part in the study to withdraw their data before they will be unable to do so.
4.	Are measures in place to provide confidentiality for participants and ensure secure management and disposal of data collected from them?	Yes	Data will be kept securely on university approved machines. Any data collected will be deleted at the end of the project.

CHECKLIST QUESTIONS		Yes/No	Explanation
5.	Does the study involve people who are particularly vulnerable or unable to give informed consent (eg, children or people with learning difficulties)?	No	
6.	Could your research cause stress, physical or psychological harm to humans or animals, or environmental damage?	No	
7.	Could any aspects of the research lead to unethical behaviour by participants or researchers (eg, invasion of privacy, deceit, coercion, fraud, abuse)?	No	
8.	Does the research involve the NHS or collection or storage of human tissue (includes anything containing human cells, such as saliva and urine)?	No	

Your explanations should indicate briefly for Qs 2-4 how these requirements will be met, and for Qs 5-8 what the pertinent concerns are.

- **Minimal Risk:** If **Q 1 is answered 'No'**, then no ethics approval is needed.
- **Low Risk:** If **Qs 2-4 are answered 'Yes' and Qs 5-8 are answered 'No'**, then no approval is needed from the *Faculty Research Ethics Committee* (FREC). However, your supervisor must approve (a) your information and consent forms (Qs 2 & 3) and (b) your measures for participant confidentiality and secure data management (Q4).
- **High Risk:** If **any of Qs 5-8 are answered 'Yes'**, then you must submit an application for full ethics approval *before* the project can start. This can take up to 6 weeks. Consult your supervisor about how to apply for full ethics approval.

Risk Assessment: Separate guidance on risk assessment can be found on UWE's Health and Safety forms webpage at https://go.uwe.ac.uk/RiskAssessment. If needed, you must complete a Risk Assessment form. This must also be attached to your application for full ethics approval if your project is **High Risk**.

Your supervisor must check your responses above before you submit this form.
Submit this completed form via the *Assignments* area in Blackboard (or elsewhere if so directed by the module leader or your supervisor).
After you have uploaded this form, your supervisor will confirm it has been correctly completed by "marking" it as *Passed*/100% via the *My Grades* link on the Blackboard.

Further research ethics guidance is available at http://www1.uwe.ac.uk/research/researchethics

APPENDIX B

APPENDIX

See below for attachment of a copy of the informed consent provided to participants. document.

Title of Study:

Teaching robots perspective taking

Principal Investigator:
John Birkett
Bristol Robotics Lab
University of the West of England, Frenchay Campus
Bristol
BS14 1QY

Supervisor:
Manuel Giuliani

Purpose of research:
You are being asked to take part in a research study. Before you take part it is important that you understand why the research is being done, what it will involve and what data will gathered from your participation. Please ask the researcher if there is anything that is not clear or if you require more information.

The purpose of this research is to use and train an Artificial Neural Network (ANN) to recognise ownership of objects, in this case numbered blocks. In order to do this a considerable amount of data must be collected to then be used to train the ANN, this data will be in the form of images. A camera will be attached in front of the robot and it will record human participants and the robots interactions as they take turns manipulating bricks.

Right to withdraw – You have until two weeks' time, following taking part, to contact the investigator to have any data collected from your participation removed. You can do this by emailing jb17969@bristol.ac.uk with your unique code. You do not need a reason to withdraw but you must do this before the aforementioned date.

Confidentiality – These images will be stored in a secure manner on a UWE encrypted device, no personal information pertaining to your identity will be stored with these images and they will not be used for any other purposes than described here. This is being insured by using a code made up of your initials, age and sex. This is linked with the trial number which you completed and is associated with the data. This will allow you to quote your individual code back in an email asking for any data to be erased.

Risks
As you will be interacting with a high powered factory line robot there can be moving parts in the working vicinity. The Yumi-ABB robot is a safe robot for human interaction, which stops moving if it comes into contact with anything. The robot will be programmed to interact in a relatively slow manner which will lower any risk of harm. You are advised not to attempt to touch the work area whilst any parts are in motion.

Benefits
There are no direct benefits to you for taking part in this research; however, we hope that this research may prove very beneficial to enhancing Human Robot Interaction in the future.

Consent
I can confirm that I am volunteering to take part in this research and I have fully read and understood the above information. I also confirm that I have been given the chance to ask questions regarding the nature of the research and have had any questions answered to a satisfactory level. I understand that I will be given a copy of this consent form. I understand that I am able to withdraw from this research, without giving reason, by the 20th July 2018 but that after this date I forfeit this option.

Participant's Signature: ……

Investigator's Signature: ……

The code used below needs to be the first letter of your forename and surname, followed by your age and then the first letter of your sex. ***For example: JB26M***

Code: Trial Number: Date:

APPENDIX C

APPENDIX

See page below for attachment of risk register for project.

Risk Register

RISK	Mitigation	Likelihood	Impact	Score
Complicated project, using multiple libraries, and integrating the model could be more time consuming and prevent completion.	Speech recognition isn't necessary for project aims. Instead of using speech recognition libraries, the research will use text-based communication to interface with robot, which will reduce complexity.	3	4	12
ABB Yumi Robot could be in use at the time of data collection.	Book time with the robot for the period of data collection (refer to timeline).	2	4	8
Not fluent in TensorFlow machine learning library. Could cause implementation difficulties.	Undertake an online course for using the library.	2	2	4
Computer vision equipment could be unavailable.	Book time with computer vision equipment (suitable web camera) from technicians prior to data collection.	1	2	2

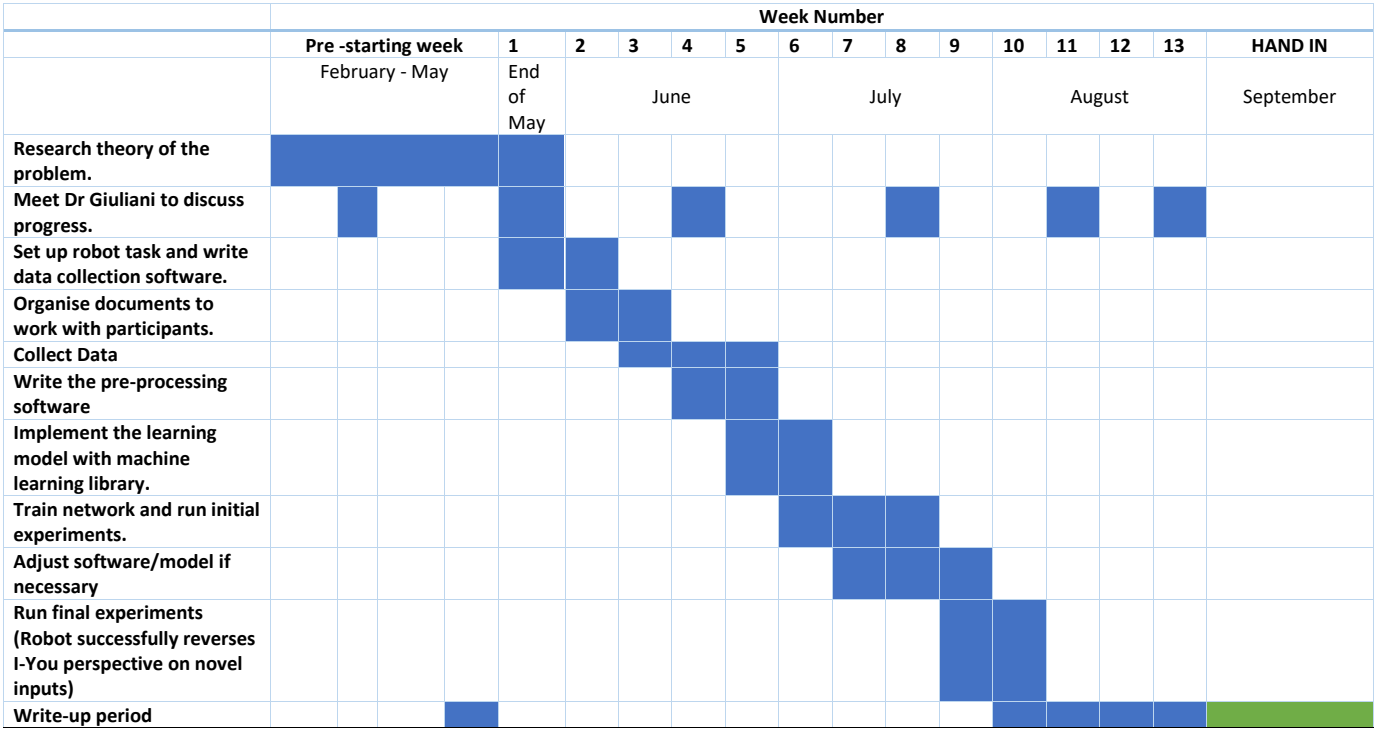
Date: 05/06/2018

APPENDIX D

APPENDIX

See below for attachment of the timeline created in May before the project commenced and an updated one created in September reflecting the true timeline of the project.

Timeline


Week Number
Pre -starting week
1
2
3
4
5
6
7
8
9
10
11
12
13
HAND IN
February - May
End of May
June
July
August
September
Research theory of the problem.
Meet Dr Giuliani to discuss progress.
Set up robot task and write data collection software.
Organise documents to work with participants.
Collect Data
Write the pre-processing software
Implement the learning model with machine learning library.
Train network and run initial experiments.
Adjust software/model if necessary
Run final experiments (Robot successfully reverses I-You perspective on novel inputs)
Write-up period

Timeline

	Week Number														
	Pre -starting week	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**HAND IN**
	February - May	End of May	June				July				August				September
Research theory of the problem.															
Meet Dr Giuliani to discuss progress.															
Set up robot task and write data collection software.															
Organise documents to work with participants.															
Collect Data															
Revision and Exam Retake															
Write the pre-processing software															
Implement the learning model with machine learning library.															
Train network and run initial experiments.															
Adjust software/model if necessary															
Evaluate Model															
Write-up period															

APPENDIX

This appendix shows the training code and evaluation code used in the methods and results section.

traincnn.py

```
import tensorflow as tf
import tensorflow.keras.preprocessing.image as img
import matplotlib.pyplot as plt
import numpy as np
import os
import tensorflow.keras as ker
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import EarlyStopping, TensorBoard, BaseLogger, CSVLogger
from sklearn.metrics import accuracy_score, f1_score
from datetime import datetime
import pandas as pd


#to be used in loading datasets
paths = ['/validation', '/test', '/train']
origin = os.getcwd()
color = 3
t_sz = 36
#Parameters
sizeimg = [t_sz, t_sz, color]
Depth = 2
colormod = 'rgb'
print("Hello")

def datagen():

  #image augmentation for training data
  train_datagen = img.ImageDataGenerator( rescale=1./255,
```

```
    rotation_range = 30,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    horizontal_flip = True,
    vertical_flip = True
    )
  # to be used for both test and validations
  test_datagen = img.ImageDataGenerator(rescale=1./255)

  #Get all images and perform data augmentation
  train_generator = train_datagen.flow_from_directory(
    directory=r"./train/",
    target_size = (t_sz,t_sz),
    color_mode = colormod,
    batch_size = 20,
    class_mode = "categorical",
    shuffle = True,
    seed = 42)

  valid_generator = test_datagen.flow_from_directory(
    directory=r"./validation/",
    target_size = (t_sz,t_sz),
    color_mode = colormod,
    batch_size = 20,
    class_mode = "categorical",
    shuffle = True,
    seed = 42)

  test_generator = test_datagen.flow_from_directory(
    directory=r"./test/",
    target_size = (t_sz,t_sz),
    color_mode = colormod,
    batch_size = 1,
    class_mode = None,
    shuffle = False,
    seed = 42)

  return train_generator, valid_generator, test_generator


def model_1(size, n_lay):
  #INPUTS IMAGE SIZE and DEEPNESS OF NET => n_lay

  filtermult = 6
  # Define mod
  model = ker.Sequential()
  model.add(Conv2D(filtermult, kernel_size = (3,3), input_shape = size, activation = 'relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.05))

  # More Conv
  model.add(Conv2D(2*filtermult, kernel_size =(3,3), activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.05))

  # model.add(Conv2D(3*filtermult, kernel_size =(3,3), activation='relu'))
  # model.add(MaxPooling2D(pool_size=(2, 2)))
  # model.add(Dropout(0.01))

  model.add(Conv2D(3*filtermult, kernel_size=(3,3), activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.05))

  model.add(Flatten())
  model.add(Dense(650, activation = 'relu'))
```

```
  #improve overfitting
  model.add(Dropout(0.5))
  # num classes is output neurons
  model.add(Dense(2, activation = 'softmax'))

    # Compile the model
  optimiz = tf.keras.optimizers.Adam(lr=0.01, beta_1=0.9, beta_2=0.999, epsilon=0.1, decay=0.0005, amsgrad=False)
  model.compile(loss='categorical_crossentropy',
    optimizer= optimiz,
    metrics=['accuracy'])
  # Print a model summary
  model.summary()

  return model

#Training HyperParameters
Epochs = 100

#Get Data
X_TRAIN, Valdata, testdata = datagen()
print(X_TRAIN)
filenames = testdata.filenames



#Tensorboard Callback
LOG_DIRECTORY_ROOT = ''
now = datetime.utcnow().strftime("%Y%m%d%H%M%S")
log_dir = "{}/run-{}/".format(LOG_DIRECTORY_ROOT, now)
tensorboard = TensorBoard(log_dir='./logs',
  histogram_freq=0, batch_size=16, write_graph=True, write_grads=True,
  write_images=True)

#Define Earlystopping parameters
Patience = 4
EarlyStop = EarlyStopping(monitor = 'val_loss', min_delta=0.001,
  patience = Patience, verbose=0, mode ='auto')

#Get Stepsizes
Step_size_train = X_TRAIN.n//X_TRAIN.batch_size
print("STEP SIZE TRAIN: " + str(Step_size_train))
Step_size_valid = Valdata.n//Valdata.batch_size
print("STEP SIZE VALID: " + str(Step_size_valid))

#batchLogCallback = LambdaCallback(on_batch_end=batchOutput)
#epoch_end = BaseLogger(stateful_metrics=None)
csv_log = CSVLogger("EpochRes", separator=',', append=False)
#Place callbacks in list
callbacks = [EarlyStop, tensorboard, csv_log]
print("Begin Training")

model1 = model_1(sizeimg, Depth)

#Train 2nd
model1.fit_generator(generator = X_TRAIN,
  steps_per_epoch = Step_size_train,
  validation_data = Valdata,
  validation_steps = Step_size_valid,
  epochs = Epochs,
  verbose = 1,
  callbacks = callbacks)

#Evaluate
model1.evaluate_generator(generator=Valdata)
#Save model
```

```
160 model1.save('my_model01.h5')
161
162 print("Finished TRAINING 2")
```

APPENDIX

The following appendix contains the programming code used for controlling the robot ABB-Yumi during data collection. The co-ordinates were gained using jogging methods, placing the robot in configurations and recording the robots joint position in sequence. This was wrote in the robots manufacturers high level language, 'RAPID'.

Left Arm

```
1  MODULE MainModule
2    TASK PERS tooldata jostnobi:=[TRUE
       ,[[19.8088,57.0777,130.185],[0.960028,0.0298825,0.278168,-0.00865853]],[0.23,[8.2,11.7,52],[1,0,0,0],0.00021,0.00024,9
       E-05]];
3
4      PROC main()
5          g_Calibrate;
6          g_GripOut;
7          break;
8      MoveJ [[553.70,332.83,307.79],[0.0845007,0.224936,-0.869693,0.431159],[-1,0,-1,4],[-132.47,9E+09,9E+09,9E+09,9
        E+09,9E+09]], v300, z50, jostnobi;
9      MoveJ [[442.65,156.28,150.74],[0.233603,-0.632776,-0.72353,-0.146729],[-1,1,0,4],[-117.646,9E+09,9E+09,9E+09,9
        E+09,9E+09]], v300, z50, jostnobi;
10     MoveJ [[426.74,163.66,24.67],[0.273228,-0.666328,-0.677123,-0.151186],[0,0,0,0],[-94.2385,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v300, z50, jostnobi;
11     MoveJ [[432.21,161.47,7.13],[0.257221,-0.675917,-0.672757,-0.156118],[0,0,0,1],[-89.1478,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v30, z50, jostnobi;
12     WaitTime 1.5;
13         !addcomment
14     g_GripIn; !Grip Number 4
15     WaitTime 1.5;
16     MoveJ [[444.66,170.44,54.86],[0.22724,-0.629192,-0.733572,-0.119801],[0,0,0,1],[-97.9841,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v20, z50, jostnobi;
17     MoveJ [[511.61,177.76,86.98],[0.167386,-0.627871,-0.737031,-0.185866],[0,0,0,1],[-100.65,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobi;
```

```
    MoveJ [[439.26,229.43,75.25],[0.270264,-0.50131,-0.810689,-0.135754],[0,-1,1,1],[-98.413,9E+09,9E+09,9E+09,9E+09,9E+09]], v20, z50, jostnobi;
    MoveJ [[399.71,160.65,56.43],[0.270933,-0.622507,-0.734129,-0.0116224],[0,-1,1,0],[-100.461,9E+09,9E+09,9E+09,9E+09,9E+09]], v20, z50, jostnobi;
    MoveJ [[469.15,98.13,71.41],[0.157157,-0.7362,-0.654043,-0.0744163],[0,-1,1,0],[-104.106,9E+09,9E+09,9E+09,9E+09,9E+09]], v20, z50, jostnobi;
    MoveJ [[432.21,161.47,7.13],[0.257221,-0.675917,-0.672757,-0.156118],[0,0,0,1],[-89.1478,9E+09,9E+09,9E+09,9E+09,9E+09]], v20, z50, jostnobi;
    WaitTime 3;
    g_GripOut;
        MoveJ [[437.02,167.40,76.75],[0.256102,-0.659722,-0.690372,-0.150218],[-1,0,0,0],[-95.0695,9E+09,9E+09,9E+09,9E+09,9E+09]], v300, z50, jostnobi;
    MoveJ [[453.85,522.81,213.49],[0.262603,-0.397714,-0.814573,-0.330659],[-1,1,0,4],[-107.078,9E+09,9E+09,9E+09,9E+09,9E+09]], v300, z50, jostnobi;
    break;
        WaitTime 0.5;
    MoveJ [[313.19,37.34,115.56],[0.323753,-0.655516,-0.672857,-0.112899],[-1,0,0,1],[-72.1101,9E+09,9E+09,9E+09,9E+09,9E+09]], v300, z50, jostnobi;
    MoveJ [[195.84,-21.10,26.12],[0.263795,0.735622,-0.604543,0.154272],[-1,-1,-2,1],[-81.9537,9E+09,9E+09,9E+09,9E+09,9E+09]], v300, z50, jostnobi;
    MoveJ [[195.83,-21.10,26.12],[0.263796,0.735624,-0.604541,0.15427],[-1,-1,-2,1],[-81.9537,9E+09,9E+09,9E+09,9E+09,9E+09]], v300, z50, jostnobi;
    WaitTime 1;
        g_GripIn;
        WaitTime 1;
        MoveJ [[232.83,-21.21,29.36],[0.209912,0.720807,-0.617586,0.234437],[-1,-1,-2,1],[-84.7246,9E+09,9E+09,9E+09,9E+09,9E+09]], v20, z50, jostnobi;
        MoveJ [[305.02,-2.35,31.75],[0.0329616,0.650107,-0.655781,0.382394],[-1,-1,-2,1],[-84.084,9E+09,9E+09,9E+09,9E+09,9E+09]], v20, z50, jostnobi;
        MoveJ [[165.87,-22.35,57.80],[0.332241,0.721644,-0.598555,0.102851],[-1,-1,-2,1],[-84.7524,9E+09,9E+09,9E+09,9E+09,9E+09]], v10, z50, jostnobi;
        MoveJ [[210.46,-37.26,148.73],[0.288432,0.714331,-0.568657,0.288388],[-1,-1,-2,1],[-103.169,9E+09,9E+09,9E+09,9E+09,9E+09]], v10, z50, jostnobi;
        MoveJ [[204.30,-17.25,41.10],[0.296225,0.730992,-0.60036,0.132171],[-1,-1,-1,1],[-84.9752,9E+09,9E+09,9E+09,9E+09,9E+09]], v20, z50, jostnobi;
        MoveJ [[195.83,-21.10,26.12],[0.263796,0.735624,-0.604541,0.15427],[-1,-1,-2,1],[-81.9537,9E+09,9E+09,9E+09,9E+09,9E+09]], v30, z50, jostnobi;
        WaitTime 2; !Grip Number 2
    g_GripOut;
    WaitTime 0.5;
    MoveJ [[340.86,27.27,102.83],[0.314549,-0.671434,-0.665628,-0.0847009],[-1,0,0,0],[-85.7309,9E+09,9E+09,9E+09,9E+09,9E+09]], v300, z50, jostnobi;
    MoveJ [[450.86,535.53,170.83],[0.325077,-0.466863,-0.739901,-0.359041],[-1,1,0,4],[-109.113,9E+09,9E+09,9E+09,9E+09,9E+09]], v300, z50, jostnobi;
    break;
    WaitTime 0.5;
    g_GripIn; !Point at Number 1
    MoveJ [[296.99,171.01,41.66],[0.305738,-0.636888,-0.707657,0.0109069],[-1,1,-1,4],[-127.602,9E+09,9E+09,9E+09,9E+09,9E+09]], v200, z50, jostnobi;
    MoveJ [[293.59,168.61,-1.84],[0.293821,-0.640206,-0.709495,0.0205532],[-2,1,-1,4],[-131.858,9E+09,9E+09,9E+09,9E+09,9E+09]], v20, z50, jostnobi;
    MoveJ [[293.06,170.73,39.34],[0.30215,-0.636285,-0.70921,0.0294583],[-1,1,-1,4],[-123.538,9E+09,9E+09,9E+09,9E+09,9E+09]], v10, z50, jostnobi;
    MoveJ [[303.20,159.80,5.19],[0.214227,-0.656843,-0.718508,0.0800631],[-2,1,0,5],[-126.9,9E+09,9E+09,9E+09,9E+09,9E+09]], v10, z50, jostnobi;
    MoveJ [[286.35,192.94,6.43],[0.39018,-0.447632,-0.802853,0.0530366],[-2,1,-1,4],[-128.356,9E+09,9E+09,9E+09,9E+09,9E+09]], v20, z50, jostnobi;
    MoveJ [[293.37,161.59,36.72],[0.528261,-0.490543,-0.648932,-0.2433],[-2,1,-1,4],[-115.915,9E+09,9E+09,9E+09,9E+09,9E+09]], v10, z50, jostnobi;
    MoveJ [[300.86,138.77,45.82],[0.256824,-0.723777,-0.554029,-0.321309],[-2,1,-1,4],[-115.974,9E+09,9E+09,9E+09,9E+09,9E+09]], v10, z50, jostnobi;
    MoveJ [[340.87,163.15,85.08],[0.189569,-0.687544,-0.696676,-0.0773936],[-1,1,-1,4],[-143.477,9E+09,9E+09,9E+09,9E+09,9E+09]], v10, z50, jostnobi;
    MoveJ [[560.03,505.62,201.43],[0.111578,-0.487474,-0.765988,-0.403957],[-1,1,-1,5],[-127.242,9E+09,9E+09,9E+09,9E+09,9E+09]], v300, z50, jostnobi;
```

```
    g_GripOut;
    break;
    WaitTime 0.5;
    g_GripIn; !Point at number 5
    MoveJ [[420.90,57.26,56.59],[0.109287,-0.584656,-0.797995,0.0971427],[-1,1,-1,4],[-148.049,9E+09,9E+09,9E+09,9
     E+09,9E+09]], v300, z50, jostnobi;
    MoveJ [[423.04,45.43,2.26],[0.147435,-0.593436,-0.788099,0.070684],[-2,1,-1,4],[-142.848,9E+09,9E+09,9E+09,9E
     +09,9E+09]], v10, z50, jostnobi;
    MoveJ [[417.51,41.88,2.60],[0.160834,-0.472711,-0.848607,0.174764],[-2,0,-1,4],[-142.849,9E+09,9E+09,9E+09,9E
     +09,9E+09]], v10, z50, jostnobi;
    MoveJ [[398.17,64.27,-8.84],[0.23786,-0.480746,-0.83435,0.127147],[-2,0,-1,4],[-142.694,9E+09,9E+09,9E+09,9E
     +09,9E+09]], v5, z50, jostnobi;
    MoveJ [[419.11,39.12,4.31],[0.150467,-0.471448,-0.849701,0.181945],[-2,0,-1,4],[-142.772,9E+09,9E+09,9E+09,9E
     +09,9E+09]], v5, z50, jostnobi;
    MoveJ [[435.01,21.57,20.50],[0.287118,-0.669045,-0.657979,-0.19237],[-2,1,-1,4],[-140.832,9E+09,9E+09,9E+09,9E
     +09,9E+09]], v5, z50, jostnobi;
    MoveJ [[435.43,6.73,28.68],[0.277487,-0.672288,-0.661527,-0.182789],[-2,1,-1,4],[-143.025,9E+09,9E+09,9E+09,9E
     +09,9E+09]], v10, z50, jostnobi;
    MoveJ [[450.36,4.76,23.66],[0.395737,-0.670616,-0.554047,-0.294445],[-2,1,0,4],[-147.987,9E+09,9E+09,9E+09,9E
     +09,9E+09]], v10, z50, jostnobi;
    MoveJ [[441.75,37.33,26.48],[0.347085,-0.654338,-0.648325,-0.176203],[-2,1,0,4],[-143.95,9E+09,9E+09,9E+09,9E
     +09,9E+09]], v10, z50, jostnobi;
    MoveJ [[437.55,236.86,111.23],[0.343217,-0.409196,-0.834585,-0.135009],[-2,0,-1,4],[-108.93,9E+09,9E+09,9E
     +09,9E+09,9E+09]], v300, z50, jostnobi;
    MoveJ [[478.79,531.81,159.41],[0.312378,-0.246109,-0.85195,-0.340636],[-1,0,-1,4],[-105.184,9E+09,9E+09,9E
     +09,9E+09,9E+09]], v300, z50, jostnobi;
      WaitTime 1;
      break;
        g_GripOut; !Grip Number 1
        MoveJ [[300.20,167.88,102.80],[0.105428,-0.523912,-0.838136,0.10922],[-1,1,-1,4],[-166.273,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v300, z50, jostnobi;
        MoveJ [[321.25,158.68,17.08],[0.200411,-0.676863,-0.680076,-0.19796],[-1,1,-1,4],[-162.933,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v300, z50, jostnobi;
        MoveJ [[319.43,153.68,9.63],[0.211259,-0.683291,-0.671411,-0.19414],[-1,1,-1,4],[-162.685,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v50, z50, jostnobi;
        WaitTime 1;
        g_GripIn;
        MoveJ [[321.88,125.86,69.60],[0.114117,-0.706549,-0.68854,-0.116954],[-1,1,-1,4],[-162.336,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v10, z50, jostnobi;
        MoveJ [[382.10,83.14,97.71],[0.0657522,0.75659,0.607111,0.233804],[-1,2,-1,4],[-163.282,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v20, z50, jostnobi;
        MoveJ [[382.10,83.14,97.71],[0.0657519,0.75659,0.607111,0.233804],[-1,2,-1,4],[-163.282,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v10, z50, jostnobi;
        MoveJ [[482.28,68.13,187.07],[0.184837,0.686671,0.580789,0.396234],[-1,2,-1,4],[-167.13,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v10, z50, jostnobi;
        MoveJ [[437.41,276.19,133.56],[0.0702969,-0.142568,-0.924379,-0.346781],[-1,1,-1,4],[-166.945,9E+09,9E
    +09,9E+09,9E+09,9E+09]], v40, z50, jostnobi;
        MoveJ [[331.89,180.90,93.72],[0.278047,-0.528207,-0.763014,-0.247986],[-1,1,-1,4],[-165.388,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v150, z50, jostnobi;
        MoveJ [[319.43,153.68,9.63],[0.211259,-0.683291,-0.671411,-0.19414],[-1,1,-1,4],[-162.685,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v200, z50, jostnobi;
        WaitTime 2.5;
        g_GripOut;
        MoveJ [[328.04,183.72,95.42],[0.156087,-0.663119,-0.690941,-0.241888],[-1,1,-1,4],[-166.145,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v300, z50, jostnobi;
        MoveJ [[559.08,550.23,234.06],[0.0110339,-0.47645,-0.723534,-0.499372],[-1,1,-1,5],[-164.32,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v300, z50, jostnobi;
        break; !Point at
        WaitTime 0.5;
        g_GripIn;
        MoveJ [[424.15,175.85,119.43],[0.142762,-0.662527,-0.716285,-0.166172],[-1,1,0,4],[-146.983,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v200, z50, jostnobi;
        MoveJ [[415.96,181.97,27.05],[0.100393,-0.658646,-0.743926,-0.0517769],[-1,1,0,4],[-143.787,9E+09,9E+09,9E
    +09,9E+09,9E+09]], v10, z50, jostnobi;
        MoveJ [[381.73,202.92,26.26],[0.462961,-0.265859,-0.844816,0.0356612],[-1,0,-1,4],[-155.166,9E+09,9E+09,9E
```

```
      +09,9E+09,9E+09]], v10, z50, jostnobi;
97        MoveJ [[419.39,169.01,41.46],[0.393906,-0.556652,-0.677854,-0.274755],[-2,1,-1,4],[-131.697,9E+09,9E+09,9E
      +09,9E+09,9E+09]], v10, z50, jostnobi;
98        MoveJ [[397.32,114.07,56.48],[0.0381839,0.874625,0.32687,0.355991],[-1,2,-1,4],[-128.829,9E+09,9E+09,9E
      +09,9E+09,9E+09]], v10, z50, jostnobi;
99        MoveJ [[484.41,148.57,180.43],[0.142081,0.775868,0.476615,0.388174],[-1,2,-1,4],[-151.376,9E+09,9E+09,9E
      +09,9E+09,9E+09]], v10, z50, jostnobi;
100       MoveJ [[411.83,285.96,108.12],[0.202269,0.369889,-0.814911,0.39773],[-1,0,-1,4],[-147.151,9E+09,9E+09,9E
      +09,9E+09,9E+09]], v200, z50, jostnobi;
101       g_GripOut;
102       WaitTime 1200;
103
104   RETURN;
105 ENDPROC
106 ENDMODULE
```

Right Arm

```
1  MODULE MainModule
2    TASK PERS tooldata jostnobiR:=[TRUE
       ,[[-0.711136,-0.273731,136.114],[0.963603,0.267293,-0.00471843,-0.00130883]],[0.23,[8.2,11.7,52],[1,0,0,0],0.00021,0.00024,9
       E-05]];
3    PROC main()
4      g_Calibrate;
5      WaitTime 1200;
6          break;
7      g_GripIn;
8          ! Point at Number 2
9      MoveJ [[369.20,-379.52,185.20],[0.361537,-0.689545,0.611725,-0.140037],[0,0,-2,4],[122.773,9E+09,9E+09,9E+09,9
        E+09,9E+09]], v300, z50, jostnobiR;
10     MoveJ [[279.41,-113.78,65.25],[0.584955,-0.674406,0.44866,0.041334],[0,-1,-2,4],[176.355,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v300, z50, jostnobiR;
11     MoveJ [[241.60,-12.17,24.13],[0.482519,-0.554322,0.668494,0.114099],[0,-1,-2,5],[166.425,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v200, z50, jostnobiR;
12     MoveJ [[256.90,-0.20,31.09],[0.111957,-0.526629,0.832682,-0.129496],[1,0,-2,5],[157.076,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
13     MoveJ [[214.20,-1.56,25.59],[0.268795,-0.882624,0.381792,0.0543879],[1,0,-3,5],[145.091,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
14     MoveJ [[258.39,1.56,28.45],[0.0732654,0.560857,-0.818654,-0.0993842],[1,1,-3,5],[153.881,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
15     MoveJ [[239.50,-13.93,34.67],[0.158798,-0.0629859,0.975353,0.139652],[0,1,-3,4],[148.231,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
16     MoveJ [[243.87,-8.81,32.41],[0.226634,-0.36508,-0.900873,0.0614945],[1,2,-3,4],[149.738,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
17     MoveJ [[256.44,-9.82,45.07],[0.204334,-0.358862,-0.905228,0.100142],[1,2,-3,4],[144.854,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
18     MoveJ [[263.21,3.05,48.02],[0.119408,-0.361348,-0.895552,0.230556],[1,2,-3,4],[138.165,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v100, z50, jostnobiR;
19     MoveJ [[555.57,-237.61,183.97],[0.0989876,-0.728266,-0.673732,-0.0769106],[1,3,-3,1],[94.7252,9E+09,9E+09,9E
        +09,9E+09,9E+09]], v300, z50, jostnobiR;
20     MoveJ [[431.16,-257.42,222.61],[0.130492,-0.24344,-0.77294,-0.571203],[0,0,-1,4],[143.204,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v200, z50, jostnobiR;
21
22     WaitTime 1;
23         break;
24         WaitTime 0.5;
25     g_GripOut;
26     MoveJ [[369.60,-147.34,52.20],[0.269596,-0.628969,0.70631,-0.181223],[0,0,-2,5],[142.423,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v200, z50, jostnobiR;
27     MoveJ [[373.80,-144.37,4.40],[0.219143,-0.692416,0.658307,-0.197909],[1,0,-2,5],[129.746,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v200, z50, jostnobiR;
28     WaitTime 1.5;
29         ! Grip Number 6
30     g_GripIn;
31     WaitTime 1;
```

```
32    MoveJ [[380.02,-138.02,45.92],[0.19949,-0.682183,0.674485,-0.199751],[1,0,-2,5],[124.254,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
33    MoveJ [[435.03,-138.68,50.94],[0.289595,-0.649575,0.650677,-0.266095],[1,0,-2,5],[124.291,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
34    MoveJ [[392.10,-95.93,50.03],[0.314936,-0.682307,0.64172,-0.153193],[1,0,-2,5],[124.727,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
35    MoveJ [[331.86,-157.18,46.65],[0.218514,-0.701621,0.674024,-0.0753058],[1,0,-2,5],[115.468,9E+09,9E+09,9E+09,9
       E+09,9E+09]], v10, z50, jostnobiR;
36    MoveJ [[362.61,-207.63,49.43],[0.168266,-0.683097,0.674236,-0.224657],[1,0,-2,5],[114.813,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
37    MoveJ [[373.80,-144.37,4.40],[0.219143,-0.692416,0.658307,-0.197909],[1,0,-2,5],[129.746,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v20, z50, jostnobiR;
38    WaitTime 2.5;
39    g_GripOut;
40    MoveJ [[373.27,-142.33,82.50],[0.187008,-0.703435,0.656893,-0.196719],[1,0,-2,5],[117.012,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v200, z50, jostnobiR;
41    MoveJ [[493.28,-326.67,146.25],[0.501877,-0.571302,0.625372,-0.175054],[0,0,-2,4],[148.882,9E+09,9E+09,9E+09,9
       E+09,9E+09]], v200, z50, jostnobiR;
42        break;
43        ! Point at Number 3
44    WaitTime 0.5;
45    g_GripIn;
46    MoveJ [[252.79,-165.93,76.68],[0.351276,-0.581116,0.718065,-0.152619],[0,-3,0,4],[156.562,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v200, z50, jostnobiR;
47    MoveJ [[257.10,-149.30,19.08],[0.34907,-0.620851,0.689739,-0.130212],[0,-3,0,4],[150.969,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v200, z50, jostnobiR;
48    MoveJ [[271.69,-156.42,33.46],[0.205898,-0.66482,0.717362,0.0318285],[0,-3,1,5],[131.088,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
49    MoveJ [[259.30,-145.38,36.24],[0.17276,-0.934025,0.2904,0.115841],[1,-4,1,5],[137.056,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
50    MoveJ [[248.34,-150.65,22.29],[0.297316,-0.472206,0.82472,-0.0919925],[0,-3,1,4],[132.92,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
51    MoveJ [[258.11,-167.47,27.05],[0.311046,-0.661389,0.682284,-0.0174199],[0,-3,1,4],[128.856,9E+09,9E+09,9E+09,9
       E+09,9E+09]], v10, z50, jostnobiR;
52    MoveJ [[249.54,-153.70,70.81],[0.248059,-0.547468,0.776131,-0.190698],[0,-3,0,4],[123.052,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
53    MoveJ [[249.54,-153.70,70.81],[0.248058,-0.547468,0.776131,-0.1907],[0,-3,0,4],[123.052,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
54    MoveJ [[466.16,-348.37,133.93],[0.452548,-0.731345,0.456319,-0.228269],[0,-3,0,5],[156.765,9E+09,9E+09,9E+09,9
       E+09,9E+09]], v150, z50, jostnobiR;
55    g_GripOut;
56        break;
57        ! Point at Number 5
58        WaitTime 0.5;
59    g_GripIn;
60    MoveJ [[351.40,-5.03,97.24],[0.202227,-0.671759,0.693449,-0.16423],[1,-2,0,4],[143.362,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v200, z50, jostnobiR;
61    MoveJ [[365.63,0.52,19.93],[0.194404,-0.675461,0.679468,-0.210431],[1,-2,0,4],[145.822,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v200, z50, jostnobiR;
62    MoveJ [[377.94,-1.89,34.63],[0.0958573,-0.655029,0.738536,-0.127723],[1,-2,-1,4],[147.937,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
63    MoveJ [[361.93,9.59,28.64],[0.045787,-0.762132,0.586949,-0.269349],[1,-2,0,4],[149.314,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
64    MoveJ [[353.57,-2.80,31.33],[0.221454,-0.733696,0.582937,-0.269876],[1,-2,0,4],[142.526,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
65    MoveJ [[366.29,-13.26,24.06],[0.327028,-0.637613,0.697423,-0.0101801],[1,-2,0,4],[134.204,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
66    MoveJ [[366.81,-2.33,25.05],[0.217369,-0.689513,0.66358,-0.192313],[1,-2,0,4],[140.038,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
67    MoveJ [[367.06,-2.17,26.86],[0.218431,-0.689328,0.664059,-0.190106],[1,-2,0,4],[139.83,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
68    MoveJ [[394.30,25.29,17.23],[0.0379451,0.459116,-0.841021,0.283648],[1,-2,0,4],[149.477,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
69    MoveJ [[373.46,12.70,20.02],[0.225419,0.13037,-0.931481,0.254035],[1,-1,0,4],[151.196,9E+09,9E+09,9E+09,9E
       +09,9E+09]], v10, z50, jostnobiR;
```

```
70    MoveJ [[394.17,17.11,42.60],[0.157696,0.415611,-0.882978,0.150826],[1,-1,0,4],[156.281,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
71    MoveJ [[411.49,7.98,32.17],[0.0195404,-0.407701,0.905275,-0.11779],[1,-1,0,4],[146.298,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
72    MoveJ [[374.15,10.11,28.38],[0.244049,-0.61003,0.748322,-0.0912054],[1,-2,0,4],[143.647,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
73    MoveJ [[389.37,14.42,31.02],[0.201147,-0.63009,0.728482,-0.178441],[1,-2,0,4],[148.424,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
74    MoveJ [[436.29,7.04,12.01],[0.193599,-0.762149,0.540753,-0.298721],[1,-2,-1,4],[152.917,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
75    MoveJ [[389.56,5.96,47.79],[0.144097,-0.581196,0.772231,-0.212383],[1,-2,0,4],[176.598,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
76    MoveJ [[394.91,-126.19,142.84],[0.199887,-0.573568,0.751186,-0.258429],[0,-2,0,4],[154.2,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v100, z50, jostnobiR;
77    MoveJ [[585.16,-348.48,157.93],[0.466912,-0.528636,0.61977,-0.344125],[0,-2,0,5],[143.291,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v100, z50, jostnobiR;
78    g_GripOut;
79        ! Point at Number 6
80        break;
81    WaitTime 0.5;
82    g_GripIn;
83    MoveJ [[360.11,-150.35,69.16],[0.246037,-0.65167,0.677841,-0.235211],[0,-2,-1,4],[161.816,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v200, z50, jostnobiR;
84    MoveJ [[374.07,-143.21,21.72],[0.212252,-0.688824,0.667203,-0.187911],[0,-2,-1,4],[155.063,9E+09,9E+09,9E+09,9
        E+09,9E+09]], v30, z15, jostnobiR;
85    MoveJ [[381.20,-145.99,31.60],[0.0527403,-0.710289,0.692944,-0.111967],[0,-2,-1,4],[155.717,9E+09,9E+09,9E
        +09,9E+09,9E+09]], v10, z50, jostnobiR;
86    MoveJ [[373.62,-134.01,23.09],[0.0332928,-0.837828,0.487189,-0.244095],[1,-2,-1,4],[151.551,9E+09,9E+09,9E
        +09,9E+09,9E+09]], v5, z50, jostnobiR;
87    MoveJ [[364.35,-145.44,29.28],[0.254848,-0.731499,0.566655,-0.280827],[0,-2,0,4],[143.558,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
88    MoveJ [[371.64,-155.67,29.31],[0.374159,-0.669132,0.639545,-0.057002],[0,-2,0,5],[140.189,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
89    MoveJ [[373.34,-144.29,21.31],[0.231039,-0.653412,0.68655,-0.219824],[0,-2,0,4],[144.165,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
90    MoveJ [[340.83,-135.46,42.76],[0.314468,-0.354404,0.876359,0.0866144],[0,-2,0,5],[142.967,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
91    MoveJ [[382.89,-139.17,21.09],[0.573841,-0.801018,-0.0204076,0.169294],[1,-3,-1,5],[142.116,9E+09,9E+09,9E
        +09,9E+09,9E+09]], v10, z50, jostnobiR;
92    MoveJ [[536.11,-387.78,139.47],[0.421799,-0.616579,0.523263,-0.410014],[0,-3,0,4],[158.984,9E+09,9E+09,9E+09,9
        E+09,9E+09]], v100, z50, jostnobiR;
93    g_GripOut;
94        ! Grip Number 3
95        break;
96    WaitTime 0.5;
97    MoveJ [[248.58,-154.99,104.81],[0.231505,-0.646779,0.695324,-0.211204],[0,-2,0,4],[159.938,9E+09,9E+09,9E+09,9
        E+09,9E+09]], v200, z50, jostnobiR;
98    MoveJ [[269.47,-147.04,4.23],[0.220936,-0.684993,0.655812,-0.227778],[0,-2,0,4],[160.334,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v200, z50, jostnobiR;
99    WaitTime 2;
100   g_GripIn;
101   WaitTime 1.2;
102   MoveJ [[277.43,-140.75,38.15],[0.214841,-0.670681,0.680284,-0.203088],[0,-2,0,4],[157.124,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
103   MoveJ [[313.80,-150.10,54.57],[0.286747,-0.644204,0.647686,-0.288584],[0,-2,0,4],[156.786,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
104   MoveJ [[273.84,-104.56,42.75],[0.283486,-0.669682,0.65513,-0.204855],[0,-2,0,4],[154.379,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
105   MoveJ [[222.66,-143.63,39.35],[0.159835,-0.713026,0.662611,-0.164297],[0,-2,0,4],[150.572,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
106   MoveJ [[339.09,-76.58,74.60],[0.290432,-0.454741,0.819165,-0.194497],[0,-2,0,4],[153.541,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v10, z50, jostnobiR;
107   MoveJ [[214.71,-58.61,84.35],[0.197788,-0.484597,0.851627,-0.0278761],[0,-2,0,4],[150.259,9E+09,9E+09,9E+09,9E
        +09,9E+09]], v20, z50, jostnobiR;
108   MoveJ [[269.47,-147.04,4.23],[0.220936,-0.684993,0.655812,-0.227778],[0,-2,0,4],[160.334,9E+09,9E+09,9E+09,9E
```

```
      +09,9E+09]], v30, z50, jostnobiR;
109     WaitTime 2.5;
110     g_GripOut;
111     MoveJ [[291.99,-148.00,87.17],[0.267799,-0.68821,0.627148,-0.247662],[0,-2,0,4],[157.488,9E+09,9E+09,9E+09,9E
          +09,9E+09]], v200, z50, jostnobiR;
112     MoveJ [[420.41,-487.48,228.52],[0.418652,-0.552393,0.659238,-0.291545],[0,-2,0,5],[129.915,9E+09,9E+09,9E+09,9
          E+09,9E+09]], v200, z50, jostnobiR;
113   ENDPROC
114 ENDMODULE
```

APPENDIX G

APPENDIX

This appendix shows the video collection code used in the methods section.

DataCollect.py

```
1
2  import cv2 as cv
3  import numpy as np
4  import os
5  import random
6  import time
7  import sys
8  import winsound
9  camera = 2
10 #Input argument variables
11 trial = sys.argv[1]
12 cond = int(sys.argv[2])
13 #Beeper Settings
14 duration = 1000
15 freq = 1000
16 #video settings
17 frmcnt = 30.0
18 vidlen = (frmcnt * 30) +1
19 # board layout
20 b_con = [[6,3,4,8,9,7,5,2,1], [9,1,2,3,4,5,6,7,8], [8,9,1,2,3,4,5,6,7], [7,8,9,1,2,3,4,5,6], [6,7,8,9,1,2,3,4,5],
       [5,6,7,8,9,1,2,3,4],[4,5,6,7,8,9,1,2,3], [3,4,5,6,7,8,9,1,2], [2,3,4,5,6,7,8,9,1]]
21 #Robot Options
22 robopt = [b_con[cond][3], b_con[cond][1], b_con[cond][0], b_con[cond][4], b_con[cond][0], b_con[cond][3],b_con[
       cond][1], b_con[cond][5], b_con[cond][2], b_con[cond][4], b_con[cond][2], b_con[cond][5]]     #Right Arm
23 #User options predefined in the order they need to go, Right hand first six, Left hand last six
24 options = [b_con[cond][6], b_con[cond][7], b_con[cond][3], b_con[cond][4], b_con[cond][6], b_con[cond][3], b_con[
       cond][8], b_con[cond][7], b_con[cond][5], b_con[cond][4], b_con[cond][8], b_con[cond][5]]
25 action = ["TOUCH", "HOLD"]
26 actlist = [0,1,0,0,1,1,1,0,0,1,0,1]
27 interation = [1,2,3,4,5,6,7,8,9,10,11,12]
28 print(robopt)
```

```
for i in range(0,12):
  # Decide which action participant takes randomly
  decider = random.uniform(0, 1)
  act = action[actlist[i]]
  #Choose User Hand
  if i < 6:
    hand = "RIGHT"
  else:
    hand = "LEFT"


  print("\n \n ==================== ROBOT TURN ================")
  cur_op = robopt[i]
  outputfilename = (trial +  "_int_" + str(interation[i]) + "_R-" + str(robopt[i]) + ".avi")
  print("Robot is now touching brick number " + str(cur_op) +" ..............")
  input("Press PLAY on FlexPendant and ENTER on keyboard at same time")
  #Bepp when beginning
  winsound.Beep(freq, duration)
  time.sleep(1.7)
  # open camera device /// 0 = default Camera /// 1 = 2nd Camera
  vid = cv.VideoCapture(camera)
  fourcc = cv.VideoWriter_fourcc(*'XVID')
  output = cv.VideoWriter(outputfilename, fourcc, frmcnt, (640,480))
  # X Range number of frames in multiples of FPS above
  for x in range(0,int(vidlen)):
    ret, frame = vid.read()
    output.write(frame)
    cv.imshow('Frame', frame)
    # delay for four seconds
    cv.waitKey(1)
    # Release File
  vid.release()
  output.release()
  cv.destroyAllWindows()

  print("\n \n ==================== USER TURN ================")
  cur_op = options[i]
  outputfilename = (trial + "_int_" + str(interation[i]) + "_U-" + str(options[i]) + ".avi")
  print("\n You should now begin to  now " + act + " brick number " + str(cur_op) + " with your " + hand +" hand
     ..............")
  input("\n Press ENTER on keyboard, when you are ready, \n you will have 2.5 seconds before video capture begins"
     )
  #Bepp when beginning
  winsound.Beep(freq, duration)
  time.sleep(2.5)
    #open camera device /// 0 = default Camera /// 1 = 2nd Camera
  vid = cv.VideoCapture(camera)
  fourcc = cv.VideoWriter_fourcc(*'XVID')
  output = cv.VideoWriter(outputfilename, fourcc, frmcnt, (640,480))
  # X Range number of frames in multiples of FPS above
  for x in range(0,int(vidlen)):
    ret, frame = vid.read()
    output.write(frame)
    cv.imshow('Frame', frame)
    # delay for four seconds
    cv.waitKey(1)
  #Beep when done
  winsound.Beep(freq, duration)
  #Release File
  vid.release()
  output.release()
  cv.destroyAllWindows()
```

APPENDIX

The following Appendix contains the code used to perform an interaction with the model.

stats.py

```
1
2
3  import tensorflow.keras.preprocessing.image as img
4
5  import numpy as np
6  import os
7  import os.path as path
8  from scipy import ndimage
9  import imageio as ii
10 import matplotlib.pyplot as plt
11 from skimage.transform import resize
12 import visvis as vv
13 import matplotlib.image as mpimg
14 import tensorflow.keras.models as models
15 import tensorflow as tf
16 import tensorflow.keras.preprocessing.image as img
17 import matplotlib.pyplot as plt
18 import numpy as np
19 import os
20 import tensorflow.keras as ker
21 from tensorflow.keras.models import Sequential
22 from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D
23 from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, confusion_matrix
24 from datetime import datetime
25 import pandas as pd
26 from imgaug import augmenters as iaa
27
28 # model = models.load_model('my_model01.h5')
29 model  = 0
30 def close_event():
31   plt.close()  #timer lls this function after 3 seconds and closes the window
32
33 #function allowing user to interact with model
34 def interact(reSize, model):
35   origin = os.getcwd()
36   Data_Path = origin + '/Test/x'
37   imagelist = os.listdir(Data_Path)
```

```
38
39    n_im = len(imagelist)
40    os.chdir(Data_Path)
41    print(imagelist)
42
43
44
45    for i in range(n_im):
46      #Get Image from DataPath directory
47      x=mpimg.imread(imagelist[i])
48      #turn pixel to float float
49      x = x/255
50
51
52      #resize image to reSize value
53      x = resize(x, (reSize, reSize), anti_aliasing=True)
54      #add extra dimension to start so shape fits in CNN
55      x = np.expand_dims(x, axis=0)
56      #print(x.shape)
57
58      #fig = plt.figure()
59      fig, axarr = plt.subplots(2,1)
60      timer = fig.canvas.new_timer(interval = 5000) #creating a timer object
61      timer.add_callback(close_event)
62
63      #Get actual class condition
64      classlabel = imagelist[i]
65      classlabel = classlabel[0]
66
67      print("classlabel = " + classlabel )
68
69      pred = model.predict(x, batch_size=None, verbose=1, steps=None)
70      predictionbinary = np.round(pred)
71      IF STATEMENTS CONTROLLING INTERACTION
72      if predictionbinary == 1:
73        humans_question = "What am I holding?"
74        predictionclass = "U"
75        robot_response = "You are holding a brick"
76      elif predictionbinary == 0:
77        humans_question = "What are you holding?"
78        predictionclass = "R"
79        robot_response = "I am holding a brick"
80       print("\n Question From User:    " + humans_question)
81      axarr[0].imshow(x[0,:,:,:])
82      axarr[1].imshow(x_aug[0,:,:,:])
83      plt.xlabel('Image: ' + str(i+1) +' of ' + str(n_im))
84      timer.start()
85      plt.show()
86      # print("\n Robots response to user:     " + robot_response)
87      input("\n \n Press Enter To Go To Next Image")
88      #print(np.round(pred))
89
90      #print(predictionclass + "the brick")
91
92
93    os.chdir(origin)
94
95  reSize = 62
96  interact(reSize, model)
```

APPENDIX I

APPENDIX

The following Appendix contains the code used to collect the statistical analysis following training of the models.

stats.py

```
import tensorflow.keras.preprocessing.image as img
import numpy as np
import os
import os.path as path
from scipy import ndimage
import imageio as ii
import matplotlib.pyplot as plt
from skimage.transform import resize
import visvis as vv
import matplotlib.image as mpimg
import tensorflow.keras.models as models
import tensorflow as tf
import tensorflow.keras.preprocessing.image as img
import matplotlib.pyplot as plt
import numpy as np
import os
import tensorflow.keras as ker
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D
from datetime import datetime
import pandas as pd
from imgaug import augmenters as iaa

originn = os.getcwd()
model = models.load_model('my_model01.h5')
Data_Path = originn + '/Test/Test'
n_conditions = 2
reSize = 36
origin = os.getcwd()
def get_test_names(Data_Path):
  imagelist = os.listdir(Data_Path)
  n_img = len(imagelist)
  return imagelist, n_img
def get_image(imagelist, num, Data_Path, reSize):
  origin = os.getcwd()
```

```
  os.chdir(Data_Path)
  x=mpimg.imread(imagelist[num])
  x = resize(x, (reSize, reSize), anti_aliasing=True)
  os.chdir(origin)
  return x
def output2csv(y_true, y_pred_Robot, y_pred_Human, imname, f1, TPR, FPR):
  df = pd.DataFrame(data={"Filename": imname, "robot owns": y_pred_Robot,
   "human owns": y_pred_Human, "actual class position": y_true,
   "f1": f1, "TPR": TPR, "FPR": FPR})
  df.to_csv("./datafile.csv", sep=',',index=False)
def get_counts(y_true, y_pred_Robot, y_pred_Human):
  TP = 0
  TN = 0
  FP = 0
  FN = 0
  for i in range(len(y_true)):
    if y_true[i] == 0:
      if y_pred_Robot[i] >=0.95:
        TP += 1
      else:
        FN += 1
      if y_pred_Human[i] >=0.50:
        FP += 1
      else:
        TN += 1
    elif y_true == 1:
      if y_pred_Human[i] >=0.95:
        TP += 1
      else:
        FN += 1
      if y_pred_Robot[i] >=0.50:
        FP += 1
      else:
        TN += 1
  return TP, TN, FP, FN
def f1(TP, TN, FP, FN):
  f1score = (2*TP)/ ((2*TP) + FP + FN)
  return f1score
def TPR_FPR(TP, TN, FP, FN):
  TPR = TP / (TP + FN)
  precision = TP / (TP + FP)
  return TPR, precision
def prediction(model, Data_Path, reSize):
  #get the image filenames
  imagelist, n_img = get_test_names(Data_Path)
  imname = []
  y_true = []
  y_pred_Robot = []
  y_pred_Human = []
  for n in range(n_img):
    #get image
    im = get_image(imagelist, n, Data_Path, reSize)
    #add extra dimension to start so shape fits in CNN
    im = np.expand_dims(im, axis=0)
    #append label to list
    name = imagelist[n]
    #append label position in numpy prediction
    imname.append(imagelist[n])
    if name[0] == 'R':
      classlabel = 0
    elif name[0] == 'U':
      classlabel = 1
    y_true.append(classlabel)
    #predict ownership
    pred = model.predict(im, batch_size=None, verbose=1, steps=None)
```

```
101     #append the predictions to robot and human class
102     y_pred_Robot.append(pred[0,0])
103     y_pred_Human.append(pred[0,1])
104   TP, TN, FP, FN = get_counts(y_true, y_pred_Robot, y_pred_Human)
105   TPR, precision = TPR_FPR(TP, TN, FP, FN)
106   f1_scr = f1(TP, TN, FP, FN)
107   output2csv(y_true, y_pred_Robot, y_pred_Human, imname, f1_scr, TPR, precision)
108 prediction(model, Data_Path, reSize)
```

References

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G. and Isard, M. [2016], Tensorflow: a system for large-scale machine learning., *in* 'OSDI', Vol. 16, pp. 265–283.

[2] Bartneck, C., Bleeker, T., Bun, J., Fens, P. and Riet, L. [2010], 'The influence of robot anthropomorphism on the feelings of embarrassment when interacting with robots', *Paladyn, Journal of Behavioral Robotics* **1**(2), 109–115.

[3] Beattie, P. F., Pinto, M. B., Nelson, M. K. and Nelson, R. [2002], 'Patient satisfaction with outpatient physical therapy: instrument validation', *Physical Therapy* **82**(6), 557–565.

[4] Bell, R. M. and Koren, Y. [2007], 'Lessons from the netflix prize challenge', *Acm Sigkdd Explorations Newsletter* **9**(2), 75–79.

[5] Bruner, J. S., Goodnow, J. J. and Austin, G. A. [1956], 'A study of thinking. new york: Science editions'.

[6] Buduma, N. and Locascio, N. [2017], *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*, " O'Reilly Media, Inc.".

[7] Card, S. K. [2017], *The psychology of human-computer interaction*, CRC Press.

[8] Chang, P.-C. [2012], 'A novel model by evolving partially connected neural network for stock price trend forecasting', *Expert Systems with Applications* **39**(1), 611–620.

[9] Chollet, F. [2015], *Keras* .

[10] Chomsky, N. [2002], *Syntactic structures*, Walter de Gruyter.

[11] Coolican, H. [2017], *Research methods and statistics in psychology*, Psychology Press.

[12] Dawson, D. L., Barnes-Holmes, D., Gresswell, D. M., Hart, A. J. and Gore, N. J. [2009], 'Assessing the implicit beliefs of sexual offenders using the implicit relational assessment procedure: A first study', *Sexual Abuse* **21**(1), 57–75.

[13] Dube, W. V., McIlvane, W. J., Callahan, T. D. and Stoddard, L. T. [1993], 'The search for stimulus equivalence in nonverbal organisms', *The Psychological Record* **43**(4), 761–778.

[14] Duffy, B. R. [2003], 'Anthropomorphism and the social robot', *Robotics and autonomous systems* **42**(3-4), 177–190.

[15] Dymond, S., May, R. J., Munnelly, A. and Hoon, A. E. [2010], 'Evaluating the evidence base for relational frame theory: A citation analysis', *The Behavior Analyst* **33**(1), 97–117.

[16] Eigen, D., Rolfe, J., Fergus, R. and LeCun, Y. [2013], 'Understanding deep architectures using a recursive convolutional network', *arXiv preprint arXiv:1312.1847* .

[17] Epley, N., Waytz, A. and Cacioppo, J. T. [2007], 'On seeing human: a three-factor theory of anthropomorphism.', *Psychological review* **114**(4), 864.

[18] Estes, W. K. and Skinner, B. F. [1941], 'Some quantitative properties of anxiety.', *Journal of experimental psychology* **29**(5), 390.

[19] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M. and Thrun, S. [2017], 'Dermatologist-level classification of skin cancer with deep neural networks', *Nature* **542**(7639), 115.

[20] Fasoli, S. E., Krebs, H. I., Stein, J., Frontera, W. R., Hughes, R. and Hogan, N. [2004], 'Robotic therapy for chronic motor impairments after stroke: follow-up results1', *Archives of Physical Medicine and Rehabilitation* **85**(7), 1106–1111.

[21] Feil-Seifer, D. and Mataric, M. J. [2005], Defining socially assistive robotics, *in* 'Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on', IEEE, pp. 465–468.

[22] Floreano, D. and Mattiussi, C. [2008], *Bio-inspired artificial intelligence: theories, methods, and technologies*, MIT press.

[23] Fukushima, K. and Miyake, S. [1982], *Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition*, Competition and cooperation in neural nets, Springer, pp. 267–285.

[24] Galizio, M. [1979], 'Contingency-shaped and rule-governed behavior: Instructional control of human loss avoidance', *Journal of the experimental analysis of behavior* **31**(1), 53–70.

[25] Garber, M. [2013], 'Funerals for fallen robots'.

[26] Girshick, R., Donahue, J., Darrell, T. and Malik, J. [2014], Rich feature hierarchies for accurate object detection and semantic segmentation, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 580–587.

[27] Graham, B. [2014], 'Fractional max-pooling', *arXiv preprint arXiv:1412.6071* .

[28] Greenway, D. E., Sandoz, E. K. and Perkins, D. R. [2010], Potential applications of relational frame theory to natural language systems, *in* 'Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on', Vol. 6, IEEE, pp. 2955–2958.

[29] Hayes, S. C. [1989], 'Nonhumans have not yet shown stimulus equivalence', *Journal of the experimental analysis of behavior* **51**(3), 385–392.

[30] Hayes, S. C. [1991], 'A relational control theory of stimulus equivalence', *Dialogues on verbal behavior* pp. 19–40.

[31] Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A. and Seybold, B. [2017], Cnn architectures for large-scale audio classification, *in* 'Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on', IEEE, pp. 131–135.

[32] Hornik, K. [1991], 'Approximation capabilities of multilayer feedforward networks'. ID: 271125.

[33] Houwer, J. D. [2013], *Advances in relational frame theory: Research and application*, New Harbinger Publications.

[34] Hubel, D. H. and Wiesel, T. N. [1962], 'Receptive fields, binocular interaction and functional architecture in the cat's visual cortex', *The Journal of physiology* **160**(1), 106–154.

[35] Jackson, F. and Hooper, N. [2017], 'Spoken rules'.

[36] Jang, H. and Lee, J. [2018], 'An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information', *IEEE Access* **6**, 5427–5437.

[37] Jorda, M., Miolane, N. and Ng, A. [2015], 'Emotion classification on face images'.

[38] Katevas, N. I., Sgouros, N. M., Tzafestas, S. G., Papakonstantinou, G., Beattie, P., Bishop, J., Tsanakas, P. and Koutsouris, D. [1997], 'The autonomous mobile robot senario: a sensor aided intelligent navigation system for powered wheelchairs', *IEEE Robotics and Automation Magazine* **4**(4), 60–70.

[39] Kim, Y. [2014], 'Convolutional neural networks for sentence classification', *arXiv preprint arXiv:1408.5882* .

[40] Kingma, D. P. and Ba, J. [2014], 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

[41] Krizhevsky, A., Sutskever, I. and Hinton, G. E. [2012], Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

[42] Köker, R., Öz, C., Çakar, T. and Ekiz, H. [2004], 'A study of neural network based inverse kinematics solution for a three-joint robot', *Robotics and autonomous systems* **49**(3-4), 227–234.

[43] Lowe, D. G. [1999], Object recognition from local scale-invariant features, *in* 'Computer vision, 1999. The proceedings of the seventh IEEE international conference on', Vol. 2, Ieee, pp. 1150–1157.

[44] McHugh, L., Barnes-Holmes, Y. and Barnes-Holmes, D. [2004], 'Perspective-taking as relational responding: A developmental profile', *The Psychological Record* **54**(1), 115–144.

[45] Mercimek, M., Gulez, K. and Mumcu, T. V. [2005], 'Real object recognition using moment invariants', *Sadhana* **30**(6), 765–775.

[46] Mirnig, N., Stollnberger, G., Miksch, M., Stadler, S., Giuliani, M. and Tscheligi, M. [2017], 'To err is robot: How humans assess and act toward an erroneous social robot', *Frontiers in Robotics and AI* **4**, 21.

[47] Muller, A. C. and Guido, S. [2017], *Introduction to machine learning with Python: a guide for data scientists*, O'Reilly Media.

[48] Nicholson, E. and Barnes-Holmes, D. [2012], 'Developing an implicit measure of disgust propensity and disgust sensitivity: Examining the role of implicit disgust propensity and sensitivity in obsessive-compulsive tendencies'.
ID: 271805.

[49] Okada, H., Sakagami, M. and Yamakawa, H. [2005], Modeling stimulus equivalence with multi layered neural networks, *in* 'International Work-Conference on Artificial Neural Networks', Springer, pp. 153–160.

[50] O'Connor, M., Farrell, L., Munnelly, A. and McHugh, L. [2017], 'Citation analysis of relational frame theory: 2009–2016'.
ID: 282078.

[51] Perez, L. and Wang, J. [2017], 'The effectiveness of data augmentation in image classification using deep learning', *arXiv preprint arXiv:1712.04621* .

[52] Powers, D. M. [2011], 'Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation'.

[53] Robotics, S. [2017], 'Pepper robot description'.

[54] Rolls, E. T. and Deco, G. [2002], *Computational neuroscience of vision.*, Oxford university press.

[55] Rosenblatt, F. [1958], 'The perceptron: a probabilistic model for information storage and organization in the brain.', *Psychological review* **65**(6), 386.

[56] Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P. and Lillicrap, T. [2017], A simple neural network module for relational reasoning, *in* 'Advances in neural information processing systems', pp. 4967–4976.

[57] Scherer, D., Müller, A. and Behnke, S. [2010], *Evaluation of pooling operations in convolutional architectures for object recognition*, Artificial Neural Networks–ICANN 2010, Springer, pp. 92–101.

[58] Sharkey, A. and Sharkey, N. [2012], 'Granny and the robots: ethical issues in robot care for the elderly', *Ethics and Information Technology* **14**(1), 27–40.

[59] Sibert, L. E. and Jacob, R. J. [2000], Evaluation of eye gaze interaction, *in* 'Proceedings of the SIGCHI conference on Human Factors in Computing Systems', ACM, pp. 281–288.

[60] Siciliano, B. and Khatib, O. [2016], *Springer handbook of robotics*, Springer.

[61] Sidman, M. [1971], 'Reading and auditory-visual equivalences', *Journal of Speech, Language, and Hearing Research* **14**(1), 5–13.

[62] Skinner, B. [1957], 'The experimental analysis of behavior', *American Scientist* **45**(4), 343–371.

[63] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. [2014], 'Dropout: a simple way to prevent neural networks from overfitting', *The Journal of Machine Learning Research* **15**(1), 1929–1958.

[64] Stewart, I. and Barnes-Holmes, D. [2001], 'Understanding metaphor: A relational frame perspective', *The Behavior Analyst* **24**(2), 191–199.

[65] Syrdal, D. S., Dautenhahn, K., Walters, M. L. and Koay, K. L. [2008], Sharing spaces with robots in a home scenario-anthropomorphic attributions and their effect on proxemic expectations and evaluations in a live hri trial., *in* 'AAAI Fall Symposium: AI in Eldercare: New Solutions to Old Problems', pp. 116–123.

[66] Vinyals, O., Toshev, A., Bengio, S. and Erhan, D. [2015], Show and tell: A neural image caption generator, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3156–3164.

[67] Weng, S. F., Reps, J., Kai, J., Garibaldi, J. M. and Qureshi, N. [2017], 'Can machine-learning improve cardiovascular risk prediction using routine clinical data?', *PloS one* **12**(4), e0174944.

[68] Wong, S. C., Gatt, A., Stamatescu, V. and McDonnell, M. D. [2016], 'Understanding data augmentation for classification: when to warp?', *arXiv preprint arXiv:1609.08764* .

[69] Xu, B., Wang, N., Chen, T. and Li, M. [2015], 'Empirical evaluation of rectified activations in convolutional network', *arXiv preprint arXiv:1505.00853* .

[70] Yamamoto, J. and Asano, T. [1995], 'Stimulus equivalence in a chimpanzee"(pan troglodytes)"', *The Psychological Record* **45**(1), 3.